GENERATIVE MODELS AND ROBUSTNESS IN DEEP LEARNING FOR INVERSE PROBLEMS

BY

ANKIT RAJ

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2020

Urbana, Illinois

Adviser:

Professor Yoram Bresler

Abstract

Image reconstruction comprises several real-life applications such as super-resolution and inpainting as well as critical medical imaging problems like CT and MRI. Deep learning based methods have recently been demonstrated to achieve state-of-the-art results on such tasks. In this thesis, we address two important aspects related to deep-learning-based image reconstruction – *(i)* architecture design and guarantees, and *(ii)* robustness and stability.

To address the first aspect, we propose (joint work with *Yuqi Li*) a new method of deploying a GAN-based prior to solve linear inverse problems using projected gradient descent (PGD). Experiments show that our approach provides a speed-up of 60-80$\times$ over earlier GAN-based recovery methods along with better accuracy. Our main theoretical result is that if the measurement matrix is moderately conditioned on the manifold range $R(G)$ and the projector is δ-approximate, then the algorithm is guaranteed to reach $O(\delta)$ recovery error in $O(log(1/\delta))$ steps in low noise regime.

Secondly, we argue that for inverse problem solvers, one should analyze and study the effect of adversaries and robustness in the measurement-space, instead of formulating in the signal-space as in previous work. We propose to introduce an auxiliary network to generate adversarial examples, which is used in a min-max formulation to build robust image reconstruction networks. Theoretically, we show for a linear reconstruction scheme the min-max formulation results in a singular-value(s) filter regularized solution, which suppresses the effect of adversarial examples occurring because of ill-conditioning in the measurement matrix.

Furthermore, we propose to use the idea of interval-bound propagation to minimize an upper bound on the reconstruction loss, given the perturbation. We show that it is computationally more efficient and gives slightly better performance in terms of robustness than the adversarial training based method that we proposed.

To my family for their love and support.
A tribute to my late grandfather.

Acknowledgments

I have had an excellent time at the University of Illinois because of my adviser, Professor Yoram Bresler. The numerous interactions I had with him over the past two years have taught me how to conduct research. Yoram has been supportive and has given me the freedom to pursue various projects without objection. He is my primary source of getting my doubts cleared and was instrumental in helping me complete this thesis. The discussions with Prof. Bo Li developed my interest in robustness in machine learning and shaped my work during the second half of my research. I would like to thank my co-author and close friend, Yuqi Li, for her significant contribution in the theoretical study which led to one of the publications during my research. Along with Yuqi, I would like to thank my other colleagues Berk, Ufuk, and Luke for making our lab a vibrant and fun place.

Since this thesis is a detailed version of two of my recent publications – ICCV'19 and arXiv'20 (currently under submission) [1, 2] – I would like to thank the anonymous reviewers, and meta-reviewers whose feedback has helped in improving the work.

My family is the foundation that has made this work possible. I would like to express my gratitude to my amazing family for the love, support, and constant encouragement I have received over the years. My parents have given me the strength to chase my dreams and fight for them. The support from my brothers and bhabhi at each step of my life has been truly special.

I am forever thankful to my friends who have motivated and guided me over the years. This work would not have been possible without them. Special thanks to my close friends: Abhijeet, Akshayaa, Ish, Sanket, Shilpa, and Varsha.

Finally, I would like to sincerely thank the National Science Foundation for supporting my research under Grant IIS 14-47879.

Table of Contents

Chapter 1

INTRODUCTION

Image reconstruction involving the recovery of an image from indirect measurements is used in many applications, including critical applications such as medical imaging, e.g., magnetic resonance imaging (MRI), computed tomography (CT) etc. Such applications demand the reconstruction to be stable and reliable. On the other hand, in order to speed up the acquisition, reduce sensor cost, or reduce radiation dose, it is highly desirable to subsample the measurement data, while still recovering the original image. This is enabled by the compressive sensing (CS) paradigm [3, 4]. CS involves projecting a high dimensional signal $x \in \mathbb{R}^n$ to a lower-dimensional measurement $y \in \mathbb{R}^m, m \ll n$, using a small set of linear, non-adaptive frames. The noisy measurement model is:

$$y = Ax + v, A \in \mathbb{R}^{m \times n}, v \sim \mathcal{N}(0, \sigma^2 I) \tag{1.1}$$

where A is the measurement matrix. The goal is to recover the unobserved natural image x from the compressive measurement y. We cover two aspects associated with inverse problems:

1. Use of generative adversarial networks (GANs) as priors – We propose a new method of deploying a GAN-based prior to solve linear inverse problems using projected gradient descent (PGD), which provides significant speed-up compared to other GAN-based methods.

2. Robustness of deep-learning-based solvers – We propose to modify the training strategy of end-to-end deep-learning-based inverse problem solvers to improve robustness. With the changed training strategy, the proposed methods outperform conventional methods in terms of our proposed robustness metric.

1.1 Generative Models for Inverse Problems

For signal priors, natural images are often considered sparse in some fixed or learnable basis [5–12]. Instead of the sparse prior commonly adopted by the CS literature, we turn to a learned prior. Neural network-based inverse problem solvers have been explored recently [13–18]. However, many of these methods [15–18] use information about the measurement matrix A while training the network. Thus, their algorithms are limited to a particular set-up to solve specific inverse-problem and usually cannot solve other problems without retraining. A few related works [19, 20] jointly optimize the measurement matrix and recovery algorithm, again resulting in algorithms limited to a particular inverse problem and measurement matrix. Instead, in our method, the network is trained independently of A and can be generalized across different inverse problems. This aspect is shared by two other neural network-based solvers [13, 14]; however, they model the image prior only implicitly by training a denoiser or a proximal map, and perhaps for this reason, appear to require a massive quantity of training samples. Importantly, very little is known about why and when they perform well, as even if the learned proximal map is assumed to be exact, there is no theoretical convergence guarantee or bound on the recovery error.

In this work, we leverage the success of generative adversarial networks (GANs) [21–26] in modeling the distribution of data. Indeed, GAN-based priors for natural images have been successfully employed to solve linear inverse problems [27–29]. However, in related work [27], the operator A is integrated into training the GAN, limiting it to a particular inverse problem. We, therefore, focus on the recent papers [28, 29] closest to our work.

Bora et al. [28] do not have a guarantee on the convergence of their algorithm for solving the non-convex optimization problem, requiring several random initializations. Similarly, Shah et al. [29] use in the inner loop a gradient descent algorithm to solve a non-convex optimization problem with no guarantee of convergence to a global optimum. Furthermore, the conditions imposed in that work on the random Gaussian measurement matrix for convergence of their outer iterative loop are unnecessarily stringent and cannot be achieved with a moderate number of measurements. Im-

portantly, both these methods require expensive computation of the Jacobian $\nabla_z G$ of the differentiable generator $G : z \to x$ that models the prior, with respect to the latent input z. Since computing $\nabla_z G$ involves back-propagation through G at every iteration, these reconstruction algorithms are computationally expensive, and even when implemented on a GPU they are slow.

We propose a GAN-based projection network to solve compressed sensing recovery problems using projected gradient descent (PGD). In numerical experiments, we are able to reconstruct the image even with a 61× compression ratio (i.e., with less than 1.6% of a full measurement set) using a random Gaussian measurement matrix. The proposed approach provides superior recovery accuracy over existing methods, simultaneously with a 60-80× speed-up, making the algorithm useful for practical applications. We also provide theoretical results on the convergence of the reconstruction error, given that the measurement matrix A satisfies certain conditions when restricted to the range $R(G)$ of the generator. We complement the theory by proposing a method to design a measurement matrix that satisfies these sufficient conditions for guaranteed convergence. We assess these sufficient conditions for both the random Gaussian measurement matrix and the designed matrix for a given dataset. Both our analysis and experiments show that with the designed matrix, 5-10× fewer measurements suffice for a robust recovery. Because the training of the GAN and projector are decoupled from the measurement operator, we demonstrate that other linear inverse problems like super-resolution and inpainting can also be solved using our algorithm without retraining the network.

The work described in this subsection was in collaboration with *Yuqi Li* [1].

1.2 Robustness for Deep-Learning-Based Image Reconstruction

Adversarial examples for deep learning based methods have been demonstrated for several applications [30–34]. It has been shown that with minute perturbations, these networks can be made to produce unexpected results. Unfortunately, these perturba-

tions can be obtained very easily. Also, there has been a plethora of work to defend against these attacks [35–42]. Recently, few papers [43, 44] introduced adversarial attacks on image reconstruction networks. In this work, we propose an adversarial training scheme for image reconstruction deep networks to provide robustness.

Recently, deep learning-based methods [45–49] have been successful for performing image reconstruction. While these methods have achieved state-of-the-art (SOTA) performance, the networks have been found to be very unstable [43], as compared to the traditional methods. Adversarial perturbations have been shown to exist for such networks, which can degrade the quality of image reconstruction significantly. Antun et al. [43] study three types of instabilities: ***(i)*** Tiny (small norm) perturbations applied to images that are almost invisible in the original images, but cause a significant distortion in the reconstructed images. ***(ii)*** Small structural changes in the original images, that get removed from the reconstructed images. ***(iii)*** Stability with increasing the number of measurement samples. We try to address the instability (i) above.

In this work, we argue that studying the instability for image reconstruction networks in the x-space as addressed by [43] is sub-optimal, and instead we should consider perturbations in the measurement, y-space. To improve robustness, we propose to modify the training strategy using two different schemes listed below.

Adversarial Training using a generator — We introduce an auxiliary network to generate adversarial examples on the fly, which are used in a min-max formulation. This results in an adversarial game between two networks while training, similar to the generative adversarial networks (GANs) [21, 50]. However, since the goal here is to build a robust reconstruction network, we make some changes in the training strategy compared to GANs. Our theoretical analysis for a special case of a linear reconstruction scheme shows that the min-max formulation results in a singular-value filter regularized solution, which suppresses the effect of adversarial examples.

Our experiment using the min-max formulation with a learned adversarial example generator for a linear reconstruction network shows that the network indeed converges to the solution obtained theoretically. For a complex non-linear deep network,

our experiments show that training using the proposed formulation results in more robust network, both qualitatively and quantitatively, compared to other methods. Further, we experimented and analyzed the reconstruction for two different measurement matrices, one well-conditioned, and another relatively ill-conditioned. We find that the behavior in the two cases is qualitatively different.

Interval-Bound Propagation (IBP) — In this approach, we try to resolve a common issue of non-convergence for the min-max formulation arising in adversarial training. The inner maximization problem is solved using gradient descent on the objective function. Since the problem is non-convex, the result obtained is non-optimal, hence we obtain a lower bound on the max, rather than the true max. Since the outer minimization tries to minimize the lower bound of the max, mitigation of the worst-case perturbation is not guaranteed. Here, we propose a method to obtain an upper bound on an inner max whose minimization will account for handling the worst-case perturbation and may be therefore preferable to adversarial training. The upper bound is obtained by the propagation of the bounds on each layer of the deep network, given the permissible perturbation on the input. Even though the upper bound obtained is quite loose, it has a significant computational advantage over adversarial training as it requires just two forward passes for the reconstruction network and does not require training of an additional generator network, as in the adversarial training set-up. In our limited experiments, we found that IBP-based training gives a more adversarially robust model compared to the adversarially trained models.

This thesis is the detailed version of two of the authors recent research works [1, 2] published during pursuit of the master of science degree:

1. *Ankit Raj*, Yuqi Li* and Yoram Bresler*, "GAN-based projector for faster recovery with convergence guarantees in linear inverse problems," in Proceedings of the IEEE International Conference on Computer Vision, 2019 – [1]

2. *Ankit Raj, Yoram Bresler, and Bo Li*, "Improving robustness of deep-learning-based image reconstruction,"arXiv preprint arXiv:2002.11821, 2020 – [2]

*Equal Contribution

Chapter 2

GAN-BASED PRIORS

2.1 Motivation

Ill-posed inverse problems require prior information on the signals for guaranteed and unique recovery. For signal priors, natural images are often considered sparse in some fixed or learnable basis [5–12]. Instead of the sparse prior commonly adopted by inverse problems literature, we turn to a learned prior. In this work, we leverage the success of the generative adversarial network (GAN) [21–26] in modeling the distribution of data. Indeed, GAN-based priors for natural images have been successfully employed to solve linear inverse problems [27–29]. However, these approaches suffer from certain issues, which we try to address in this work described below.

This work was a collaboration with a graduate student colleague *Yuqi Li* [1].

2.2 Related Work

In a recent work [27], the operator A is integrated into training the GAN, limiting it to a particular inverse problem. We, therefore, focus on the recent papers [28,29] closest to our work, for extensive comparisons. Bora et al. [28] do not have a guarantee on the convergence of their algorithm for solving the non-convex optimization problem, requiring several random initializations. Similarly, in another related work [29], the inner loop uses a gradient descent algorithm to solve a non-convex optimization problem with no guarantee of convergence to a global optimum. Furthermore, the conditions imposed by Shah et al. [29] on the random Gaussian measurement matrix for convergence of their outer iterative loop are unnecessarily stringent and

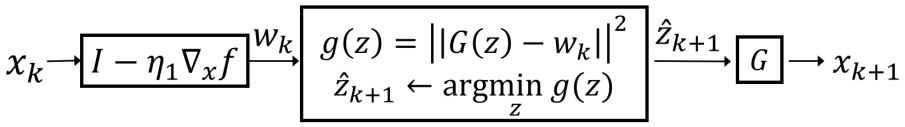
cannot be achieved with a moderate number of measurements. Importantly, both these methods require expensive computation of the Jacobian $\nabla_z G$ of the differentiable generator $G : z \to x$ that models the prior, with respect to the latent input z. Since computing $\nabla_z G$ involves back-propagation through G at every iteration, these reconstruction-algorithms are computationally expensive and even when implemented on a GPU they are slow.

2.3 Problem Formulation

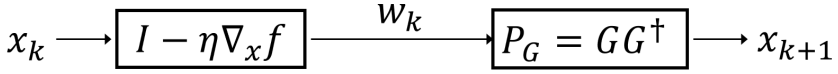
Let $x^* \in \mathbb{R}^n$ denote a ground truth image, A a fixed measurement matrix, and $y = Ax^* + v \in \mathbb{R}^m$ the noisy measurement, with noise $v \sim \mathcal{N}(0, \sigma^2 I)$. We assume that the ground truth images lie in a non-convex set $S = R(G)$, the range of generator G. The maximum likelihood estimator (MLE) of x^*, $\hat{x}_{MLE}$, can be formulated as follows:

$$\hat{x}_{MLE} = \underset{x \in R(G)}{\arg\min} - \log p(y|x) \quad = \underset{x \in R(G)}{\arg\min} \|y - Ax\|_2^2 \tag{2.1}$$

Bora et al. [28] (whose algorithm we denote by CSGM) solve the optimization problem $\hat{z} = \arg\min_{z \in \mathbb{R}^k} \|y - AG(z)\|^2 + \lambda \|z\|^2$ in the latent space (z), and set $\hat{x} = G(\hat{z})$. Their gradient descent algorithm often gets stuck at local optima. Since the problem is non-convex, the reconstruction is strongly dependent on the initialization of z and requires several random initializations to converge to a good point. To resolve this problem, Shah and Hegde [29] proposed a projected gradient descent (PGD)-based method (which we call PGD-GAN) to solve (2.1), shown in Fig. 2.1(a). They perform gradient descent in the ambient (x)-space and project the updated term onto $R(G)$. This projection involves solving another non-convex minimization problem (shown in the second box in Fig. 2.1(a)) using the Adam optimizer [51] for 100 iterations from a random initialization. No convergence result is given for this iterative algorithm to perform the non-linear projection and the convergence analysis for the PGD-GAN algorithm [29] only holds if one assumes that the inner loop succeeds in finding the optimum projection.



(a) PGD with inner-loop



(b) Network-based PGD (NPGD)

Figure 2.1: (a) Block diagram of PGD using inner-loop [29]. k represents the outer loop iterators and $\hat{z}_{k+1}$ is the optimizer of $\|G(z) - w_k\|^2$ obtained by solving the inner-loop using Adam optimizer. (b) Block diagram of our network-based PGD (NPGD) with $P_G = GG^\dagger$ as a network based projector onto $R(G)$.
$f(x) = \|Ax - y\|^2$ is the cost function defined in (2.1).

2.4 Network-Based Projector

In this section, we introduce our methodology and architecture to train a network-based projector using two methods: (i) generator-based projector using a pre-trained generator G and, (ii) end-to-end projector using a pre-trained discriminator D and how we use this projector to obtain the optimizer in (2.1).

2.4.1 Inner-Loop-Free Scheme

We show that by carefully designing a network architecture with a suitable training strategy, we can train a projector onto $R(S)$, the range of the set S which defines the signals to be recovered, thereby removing the inner-loop required in the earlier approach. The resulting iterative updates of our network-based PGD (NPGD) algorithm are shown in Fig. 2.1(b). This approach eliminates the need to solve the non-convex optimization problem in the inner-loop, which depends on initialization and requires several restarts. Furthermore, our method provides a significant speed-up by a factor of 30-40$\times$ on the CelebA dataset for two major reasons: (i) since there is no inner-loop, the total number of iterations required for convergence is significantly

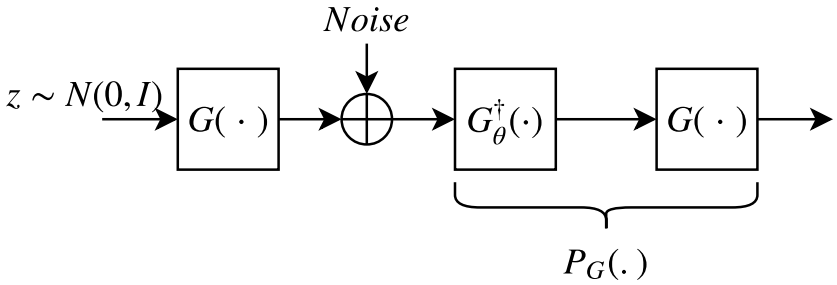


Figure 2.2: Architecture to train a Generator(G)-based projector.

reduced, (ii) does not require computation of ∇G_z i.e. the Jacobian of the generator with respect to the input, z. This expensive operation repeats back-propagation through the network for $T_{out} \times \#_{restarts}$(in the algorithm by Bora et al. [28]) or $T_{out} \times T_{in}$ (in the algorithm by Shah et al. [29]) times, where $\#_{restarts}, T_{out}$ and T_{in} are the number of restarts, outer and inner iterations respectively.

2.4.2 Generator-Based Projector

A GAN consists of two networks, generator and discriminator, which follow an adversarial training strategy to learn the data distribution. A well-trained generator $G : \mathbb{R}^k \to R(G) \subset \mathbb{R}^n, k \ll n$ takes in a random latent variable $z \sim \mathcal{N}(0, I_k)$ and produces sharp-looking images imitating the training data distribution in $\mathbb{R}^n$. The goal is to train a network that projects an image $x \in \mathbb{R}^n$ onto $R(G)$. The projector, P_S onto a set S should satisfy two main properties: (i) *Idempotence*, for any point x, $P_S(P_S(x)) = P_S(x)$, (ii) *Least distance*, for a point $\tilde{x}$, $P_S(\tilde{x}) = \arg\min_{x \in S} \|x - \tilde{x}\|^2$. Figure 2.2 shows the network structure we used to train a projector using the G. We define the multi-task loss to be:

$$\begin{aligned} \mathcal{L}(\theta) = \mathbb{E}_{z,\nu} \left[\left\| G\left(G_\theta^\dagger \left(G(z) + \nu \right) \right) - G(z) \right\|^2 \right] \\ + \mathbb{E}_{z,\nu} \left[\lambda \left\| G_\theta^\dagger \left(G(z) + \nu \right) - z \right\|^2 \right] \end{aligned} \tag{2.2}$$

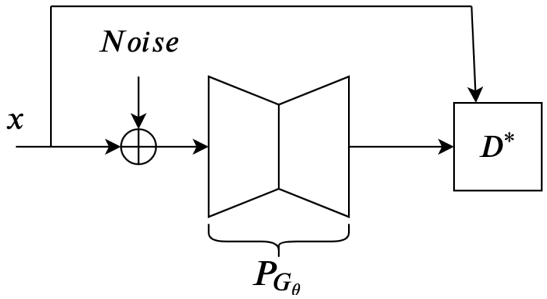


Figure 2.3: Architecture to train an end-to-end projector using fixed discriminator D^*.

where G is a generator obtained from the GAN trained on a particular dataset. Operator $G^\dagger_\theta : \mathbb{R}^n \to \mathbb{R}^k$, parameterized by θ, approximates a non-linear least squares pseudo-inverse of G and $\nu \sim \mathcal{N}(0, I_n)$ indicates noise added to the generator's output for different $z \sim \mathcal{N}(0, I_k)$ so that the projector network denoted by $P_G = GG^\dagger_\theta$ is trained on points outside the range(G) and learns to project them onto $R(G)$. The objective function consists of two parts. The first is similar to standard *Encoder-Decoder* framework; however, the loss function is minimized over θ – the parameters of $G^\dagger$, while keeping the parameters of G (obtained by standard GAN training) fixed. This ensures that $R(G)$ does not change and $P_G = GG^\dagger$ is mapping onto $R(G)$. The second part is used to keep $G^\dagger(G(z))$ close to true z used to generate training image $G(z)$. This second term can be considered a regularizer for training the projector with λ being the regularization constant.

2.4.3 End-to-End (E2E) Projector

In this formulation, we allow the parameters of both, $G^\dagger$ and G learnable, while training the projector. A well-trained discriminator $D : R^n \to R$, in a GAN set-up, takes in an image and gives the probability of the image being real. Since it learns the decision boundary between real and fake images, D^* can be used to define the set onto which the image has to be projected. $P_{G_\theta} = G(G^\dagger)$ represents the E2E projector, parameterized by θ. Figure 2.3 shows the network structure we used to

Algorithm 1 Network-based Projected Gradient Descent

Input: loss function f, $A, y, G, G^\dagger$
Parameter: step size $\eta(=\frac{1}{\beta})$
Output: an estimate $\hat{x} \in R(G)$

Let $t = 0, x_0 = A^T y$.
while $t < T$ **do**
    $w_t := x_t - \eta A^T(Ax_t - y)$
    $x_{t+1} := G(G^\dagger(w_t))$
end while
return $\hat{x} = x_T$

train a projector using the D^*. We define the multi-task loss to be:

$$\begin{aligned}\mathcal{L}(\theta) = \mathbb{E}_{x,\nu}\left[\left\|P_{G_\theta}(x+\nu) - x\right\|^2\right] \\ - \lambda \mathbb{E}_{x,\nu}\left[\log D^*\left(P_{G_\theta}(x+\nu)\right)\right]\end{aligned} \tag{2.3}$$

where D^* is a discriminator obtained from the GAN trained on a particular dataset. Operator $P_{G_\theta} = G(G^\dagger)$, approximates a projector onto the set of real images defined by the decision boundary of the D^*. The objective function consists of two parts. The first is similar to standard least-squares projection loss. The second part is the adversarial loss which tries to maximize the probability of the projected image being natural.

2.5 Theoretical Study

2.5.1 Convergence Analysis

Let $f(x) = \|Ax - y\|_2^2$ denote the loss function of projected gradient descent. Algorithm 1 describes the proposed network-based projected gradient descent (NPGD) to solve Equation (2.1).

Definition 1 (Restricted Eigenvalue Constraint (REC)) *Let* $S \subset \mathbb{R}^n$. *For*

some parameters $0 < \alpha < \beta$*, matrix* $A \in \mathbb{R}^{m \times n}$ *is said to satisfy the* $REC(S, \alpha, \beta)$ *if the following holds for all* $x_1, x_2 \in S$*.*

$$\alpha \|x_1 - x_2\|^2 \leq \|A(x_1 - x_2)\|^2 \leq \beta \|x_1 - x_2\|^2. \tag{2.4}$$

Definition 2 (Approximate Projection using GAN) *A concatenated network* $G(G^\dagger(\cdot)) : \mathbb{R}^n \to R(G)$ *is a* δ*-approximate projector, if the following holds for all* $x \in \mathbb{R}^n$*:*

$$\|x - G(G^\dagger(x))\|^2 \leq \min_{z \in \mathbb{R}^k} \|x - G(z)\|^2 + \delta \tag{2.5}$$

Theorem 1 provides upper bounds on the cost function and reconstruction error of our NPGD algorithm after n iterations.

Theorem 1 *Let matrix* $A \in \mathbb{R}^{m \times n}$ *satisfy the* $REC(S, \alpha, \beta)$ *with* $\beta/\alpha < 2$*, and let the concatenated network* $G(G^\dagger(\cdot))$ *be a* δ*-approximate projector. Then for every* $x^* \in R(G)$ *and measurement* $y = Ax^*$*, executing Algorithm 1 with step size* $\eta = 1/\beta$*, will yield* $f(x_n) \leq (\frac{\beta}{\alpha} - 1)^n f(x_0) + \frac{\beta\delta}{2-\beta/\alpha}$*. Furthermore, the algorithm achieves* $\|x_n - x^*\|^2 \leq \big(C + \frac{1}{2\alpha/\beta - 1}\big)\delta$ *after* $\frac{1}{2-\beta/\alpha} \log\big(\frac{f(x_0)}{C\alpha\delta}\big)$ *steps. When* $n \to \infty$*,* $\|x^* - x_\infty\|^2 \leq \frac{\delta}{2\alpha/\beta - 1}$*.*

Proof 1 *Please refer to the appendix.*

From Theorem 1, one important factor is the ratio β/α. This ratio largely determines the speed of linear ("geometric") convergence, as well as the reconstruction error $\|x^* - x_\infty\|^2$ at convergence. We would like β/α ratio as close to 1 as possible and must have $\beta/\alpha < 2$ for convergence. It has been shown [52] that a random matrix A with orthonormal rows will satisfy this condition with high probability for m roughly linear in dimension k with log factors dependent on the properties of the manifold, in this case, $R(G)$. However, as we demonstrate later (see Fig. 2.4), a random matrix often will not satisfy the desired condition $\beta/\alpha < 2$ for small or moderate m. To extend into such regimes, we propose next a fast heuristic method to find a relatively good measurement matrix for an image set S, given a fixed m.

2.5.2 Generator-based Measurement Matrix Design

There have been a few attempts to optimize the measurement matrix based on the specific data distribution. Hegde et al. [53] find a deterministic measurement matrix that satisfies $REC(S, 1-\delta_S, 1+\delta_S)$ for a given finite set S of size $|S|$, but their time complexity is $O(n^3 + |S|^2 n^2)$. Because the secant set S (defined later) would be of cardinality $|S| = O(M^2)$ for a training set of size M, with $M \gg n$, the time complexity would be infeasible even for fairly small n-pixel images. Furthermore, the final number of required measurements m, which is determined by the algorithm, depends on the isometry constant δ_S, and cannot be specified in advance. Kvinge et al. [54] introduced a heuristic iterative algorithm to find a measurement matrix with orthonormal rows that satisfies the REC with small β/α ratio, but their time complexity is $O\left(n^5\right)$ and the space complexity is $O(n^3)$, which is infeasible for a high-dimensional image dataset. Instead, our method, based on sampling from the secant set, has time complexity $O(Mn^2 + n^3)$, and space complexity $O(n^2)$, where M is a tiny fraction of $|S|$.

Definition 3 (Secant Set) *The normalized secant set of G is defined as follows:*

$$\mathcal{S}(G) = \left\{ \frac{x_1 - x_2}{\|x_1 - x_2\|} : x_1, x_2 \in R(G) \right\} \tag{2.6}$$

and the associated distribution is denoted as Π_S, where

$$z_1, z_2 \sim \mathcal{N}(0, I_k), s = \frac{G(z_1) - G(z_2)}{\|G(z_1) - G(z_2)\|} \sim \Pi_S \tag{2.7}$$

Given $\mathcal{S}(G)$, the optimization over A is as follows:

$$\begin{aligned} &\min_{A \in \mathbb{R}^{m \times n}} \frac{\beta}{\alpha} = \min_{A \in \mathbb{R}^{m \times n}} \frac{\max_{s \in \mathcal{S}(G)} \|As\|^2}{\min_{s \in \mathcal{S}(G)} \|As\|^2} \\ &\leq \min_{AA^T = I_m} \frac{1}{\min_{s \in \mathcal{S}(G)} \|As\|^2} = \left(\max_{AA^T = I_m} \min_{s \in \mathcal{S}(G)} \|As\|^2 \right)^{-1} \end{aligned} \tag{2.8}$$

The inequality is due to an additional constraint on $A : AA^T = I_m$. This results in the largest singular value of A being 1 and hence the numerator term, $\max_{s \in \mathcal{S}(G)} \|As\|^2$, is at most 1. As the minimization in (2.8) requires iterating through the set S, we use the expected value over $s \sim \Pi_S$ as a surrogate objective

$$A = \underset{AA^T = I_m}{\arg\max} E_{s \sim \Pi_S} \left[\|As\|^2 \right] \approx \underset{AA^T = I_m}{\arg\max} \frac{1}{M} \sum_{j=1}^{M} \|As_j\|^2 \tag{2.9}$$

The last approximation replaces the surrogate objective by its empirical estimate obtained by sampling $M \gg n$ secants $(s_j)_{j=1}^M$ according to Π_S. For m and M large enough, this designed measurement matrix would satisfy the condition $\beta/\alpha < 2$ for most of the secants in $R(G)$. Constructing an $n \times M$ matrix $D = [s_1|s_2|\ldots|s_M]$, (2.9) reduces to:

$$A^* = \underset{A}{\arg\max} \|AD\|_F^2 \text{ s.t. } AA^T = I_m \tag{2.10}$$

The optimal A^* in (2.10) has rows equal to the m leading eigenvectors DD^T. We compute $DD^T = \sum_{j=1}^{M} s_j s_j^T$ and its eigenvalue decomposition at time complexity $O(Mn^2 + n^3)$ and space complexity $O(n^2)$.

Our approach to the design of A is related to one of the steps described by Kvinge et al. [54], however by using the sampling-based estimates per (2.7) and (2.9) rather than the secant set for the entire training set, we reduce the computational cost by orders of magnitude to a modest level.

REC Histogram for A

We analyze the *REC* conditions by plotting the histogram of $\|As\|$ values for different measurement matrices $A \in R^{m \times n}$ in Fig. 2.4 where $s \in S$, the secant set of the samples from G trained on MNIST dataset. The left column shows the histograms for the random and G-based designed matrix. For random A, the spread of $\|As\|$ is clearly wider for few measurements m, resulting in $\beta/\alpha \not< 2$. For the designed A, the histogram is more concentrated. Even with as few as $m = 20$ measurements (for

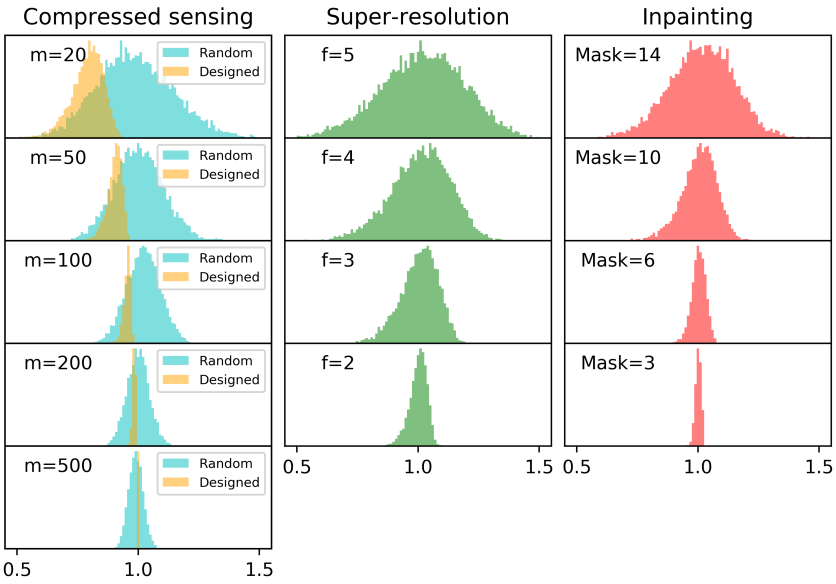


Figure 2.4: Distribution of $\|As\|$ with different A. Left: Random (cyan) and Designed matrix (orange) with different m. Middle: Downsampling matrix (green) with different f. Right: Inpainting matrix (red) with different mask size.

MNIST), the designed A satisfies the sufficient condition $\beta/\alpha < 2$ for convergence of the PGD algorithm, thus ensuring stable recovery. The middle column shows the histograms corresponding to the downsampling A that takes the spatial averages of $f \times f$, $f = 2, 3, 4, 5$, pixel values to generate low-resolution images. The right column shows the histograms for the inpainting A that masks out a centered square of various sizes. As expected, with more difficult recovery problems the spread increases. However, for each inverse problem (defined by a matrix A), the ratio β/α can be estimated for e.g., 99.9% of the samples, providing, in combination with Theorem 1, an explicit quantitative guarantee.

2.6 Experiments and Results

Detailed experiments have been done with the generator-based projector for different inverse problems as this projector gives a better theoretical understanding of

the algorithm. However, preliminary experiments with E2E projector, for which the theoretical analysis does not hold, shows that it is better than the G-based projector, empirically.

Network Architecture: We implement two GAN architectures: (i) Deep convolutional GAN (DCGAN) [55] for MNIST and CelebA, (ii) Self-attention GAN (SAGAN) [56] for LSUN church-outdoor dataset. DCGAN builds on multiple convolution, transpose convolution, and ReLU layers, and uses batch normalization and dropout for better generalization, whereas SAGAN combines convolutions with self-attention mechanisms in both, generator and discriminator, allowing for long-range dependency modeling to generate images with high-resolution details. For DCGAN, we have used the standard objective function of the adversarial loss, whereas, for SAGAN, we minimized the hinge version of the adversarial loss [57].

The architecture of the model $G^\dagger$ is similar to that of the discriminator D in the GAN and only differs in the final layer, where we add a fully connected layer with output size same as the latent variables dimension k. For training $G^\dagger$, we used the architecture shown in Fig. 2.2 and objective defined in (2.2) while keeping the pre-trained G fixed. We found that using $\lambda = 0.1$, in (2.2), gave the best performance. The noise ν used for perturbing the training images $G(z)$ follows $\mathcal{N}(0, \sigma^2 I)$. We observed that training with low σ results in a projector similar to an identity operator and hence only projecting close-by points onto $R(G)$, whereas for large σ the projector violates idempotence. We empirically set $\sigma = 1$. We then obtain a projection network $P_G = GG^\dagger$ that approximately projects images lying outside $R(G)$ onto $R(G)$. We empirically pick latent variable dimension $k = 100$.

MNIST dataset [58] consists of 28×28 greyscale images of digits with $50,000$ training and $10,000$ test samples. We pre-train the GAN consisting of 4 transposed convolution layers for G and 4 convolution layers in the discriminator D using rescaled images lying between $[-1, 1]$. We use $z \sim \mathcal{N}(0, I_k)$ as the G's input. The GAN is trained using the Adam optimizer with learning rate 0.0001, a mini-batch size of 128 for 40 epochs. For training the pseudo-inverse of G i.e. $G^\dagger$, we minimize the objective (2.2), using samples generated from $G(z)$, and with the same hyper-parameters used

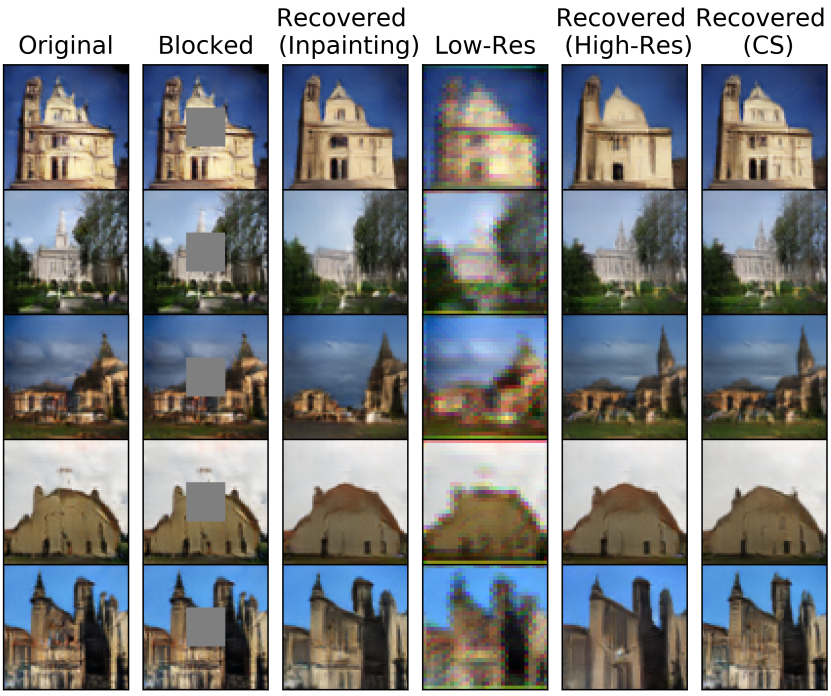


Figure 2.5: Recovery of LSUN church-outdoor images in inpainting (mask size = 20), super-resolution (4×) and compressed sensing (CS, number of measurements $m = 1000$) tasks.

for the GAN.

CelebA dataset [59] consists of more than $200,000$ celebrity images. We use the aligned and cropped version, which preprocesses each image to a size of $64 \times 64 \times 3$ and scaled between $[-1, 1]$. We randomly pick $160,000$ images for training the GAN. Images from the $40,000$ held-out set are used for evaluation. The GAN consists of 5 transposed convolution layers in the G and 5 convolution layers in D. GAN is trained for 35 epochs using Adam optimizer with learning rate 0.00015 and mini-batch size 128. $G^\dagger$ is trained in the same way as for the MNIST dataset.

LSUN church-outdoor dataset [60] consists of more than $126,000$ cropped and aligned images of size $64 \times 64 \times 3$ scaled between $[-1, 1]$. DCGAN generates high-resolution details using spatially local points in lower-resolution feature maps, whereas

in SAGAN, details can be generated using information from many feature locations making it a natural choice for diverse dataset such as LSUN. The SAGAN consists of 4 transposed convolution layers and 2 self-attention modules at different scales in G and 4 convolution layers and 2 self-attention modules in D. Each self-attention module consists of 3 convolution layers and are added at the $3rd$ and $4th$ layers of the two networks. SAGAN uses conditional batch normalization in G and projection in D. Spectral normalization is used for the layers in both G and D. We use ADAM optimizer with $\beta_1 = 0$ and $\beta_2 = 0.9$, learning rate 0.0001 and mini-batch size 64 for the GAN training. $G^\dagger$, consisting of self-attention mechanism similar to D, is trained using the objective 2.2 using the ADAM optimizer with $\beta_1 = 0.9$ and $\beta_2 = 0.999$, learning rate 0.001 and mini-batch size of 64 for 100 epochs.

We compare the performance of our algorithm on MNIST and CelebA with other GAN-prior solvers ([28, 29]) and sparsity-based methods, Lasso with discrete cosine transform (DCT) basis [61] and total variation minimization method (TVAL3) [12] for linear inverse problems, namely compressed sensing (CS), super-resolution and inpainting. For CS, we extensively evaluate the reconstruction performance with the random Gaussian and designed measurement matrices. Furthermore, we demonstrate the recovery of LSUN church-outdoor dataset images using the proposed method for the different problems in Fig. 2.5.

2.6.1 Compressed Sensing

Recovery with random Gaussian matrix

In this set-up, we use the same measurement matrix A as used in related papers [1,28,29] i.e. $A_{i,j} \sim N(0, 1/m)$ where m is the number of measurements. For MNIST, the measurement matrix $A \in R^{m\times 784}$, with $m = 20, 50, 100, 200$, whereas for CelebA, $A \in R^{m\times 12288}$, with $m = 200, 500, 1000, 2000$. Figure 2.6 shows the recovery results for MNIST images from the test set. Our NPGD algorithm performs better than others and avoids local optima. Figure 2.7 shows the reconstruction of eight test images from CelebA. Our algorithm outperforms the other three methods visually as

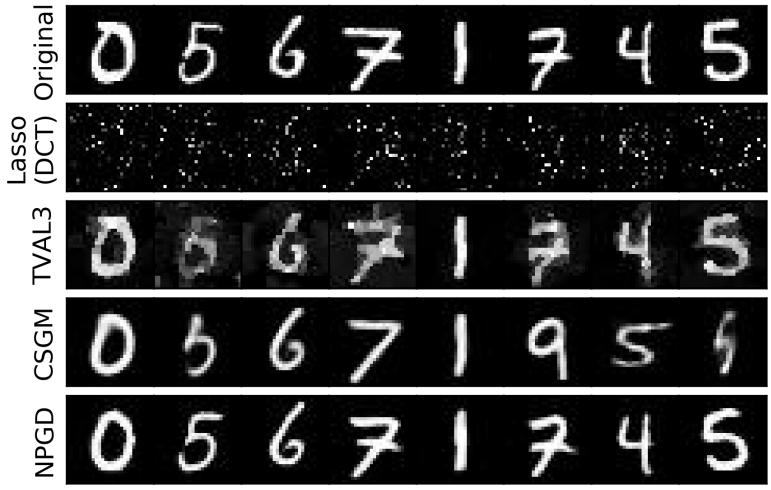


Figure 2.6: Reconstruction using Gaussian matrix with $m = 100$.

it is able to preserve detailed facial features such as sunglasses, hair and has accurate color tones. Figures 2.8(a) and 2.8(b) provide a quantitative comparison for MNIST and CelebA, respectively.

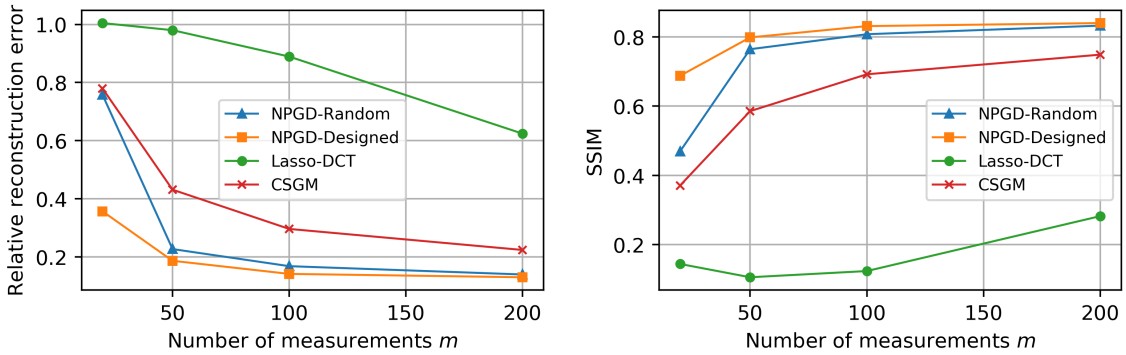
Recovery with the designed matrix

In this set-up, we use the G-based designed A described in the Section 2.5.2. We observe that recovery with the designed A is possible for much fewer measurements m. This corroborates our assessment based on Fig. 2.4 that the designed matrix satisfies the desired REC condition with high probability for most of the secants, for smaller m. Figures 2.8(a) and 2.8(b) show that our algorithm consistently outperforms other approaches in terms of reconstruction error and structural similarity index (SSIM) for a random A. Furthermore, with the designed A, we are able to get performance on-par with the random matrix using 5-10$\times$ smaller m. Figures 2.9(a) and 2.9(b) show the recovered images with the designed and a random A using our algorithm for different m. Clearly, recovery with the random A requires much bigger m than the designed one to achieve similar performance.

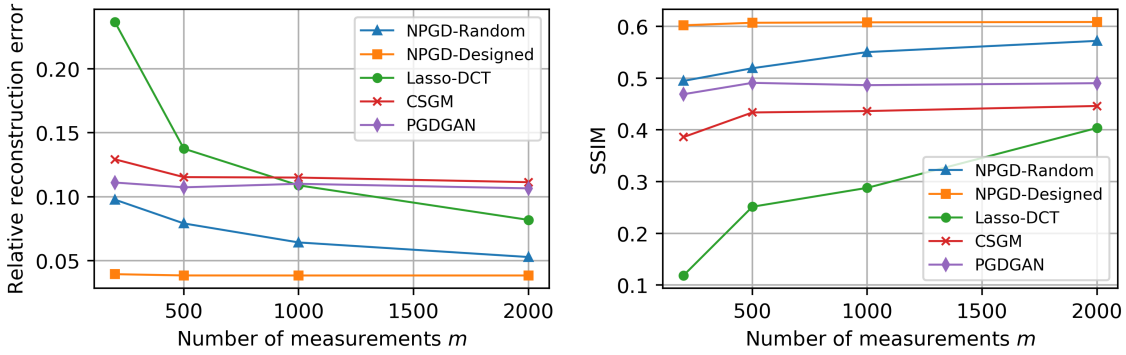
Figure 2.7: Reconstruction using Gaussian matrix with $m = 1000$.

2.6.2 Super-Resolution

Super-resolution refers to recovering the high-resolution image from a single low-resolution image, often modeled as a blurred and downsampled image of the original. This super-resolution problem is just a special case in our framework of linear measurements. We simulate the blurring+downsampling by taking the spatial averages of $f \times f$ pixel values (in RGB color space), where f is the ratio of downsampling. This corresponds to blurring by an $f \times f$ box impulse response, followed by downsampling. We test our algorithm with $f = 2, 3, 4$, corresponding to $4\times$, $9\times$ and $16\times$-smaller image sizes, respectively. We note that for higher f, the measurement matrix A may not satisfy the desired $REC(S, \alpha, \beta)$ with $\frac{\beta}{\alpha} < 2$ (see Fig. 2.4) required for convergence of our algorithm and, consequently, our theorem might not be applicable. Results for MNIST in Fig. 2.10(a)–2.10(c) show that recovery performance indeed degrades with increasing f; however, our NPGD algorithm, gives better reconstructions than Bora et al. [28].



(a)

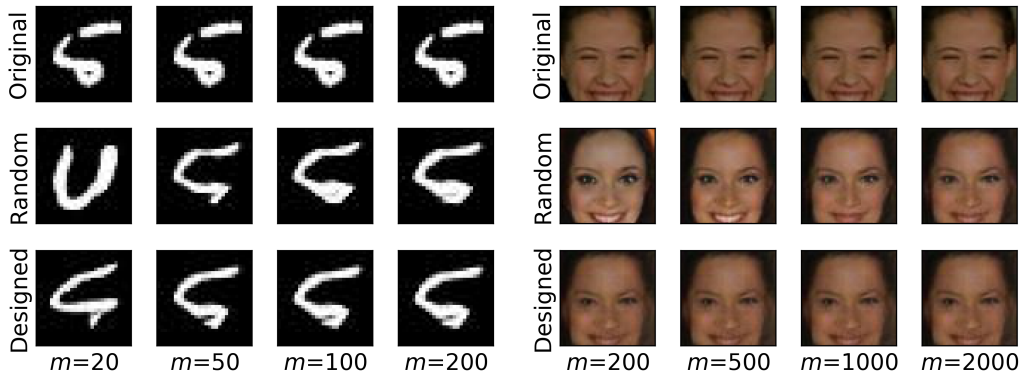


(b)

Figure 2.8: (a) Relative error $\|x^* - \hat{x}\|^2/\|x^*\|^2$ and SSIM of reconstruction algorithms for MNIST dataset with $m = 20, 50, 100, 200$ measurements. (b) Relative error and SSIM for CelebA dataset with $m = 200, 500, 1000, 2000$ measurements.

2.6.3 Inpainting

Inpainting refers to recovering the entire image from a partly occluded version. In this case, y is an image with masked regions and A is the linear operation applying a pixel-wise mask to the original image x. Again, this is a special case of linear measurements where each measurement corresponds to an observed pixel. For experiments on the MNIST dataset, we apply a centered square mask of size $6, 10, 14$. Recovery results in Fig. 2.11(a)–2.11(c) show that our method consistently outperforms the method of Bora et al. [28] and recovers almost perfectly for mask-size less than 10. The



(a) (b)

Figure 2.9: (a) MNIST reconstructions with a random Gaussian (middle row) and the designed matrix with orthonormal rows based on G (bottom row) using different m. (b) CelebA reconstructions, as in (a).

results align with the *REC* histogram for inpainting (Fig. 2.4), which shows that for higher mask-size, the desired *REC* condition for guaranteed convergence may not be satisfied.

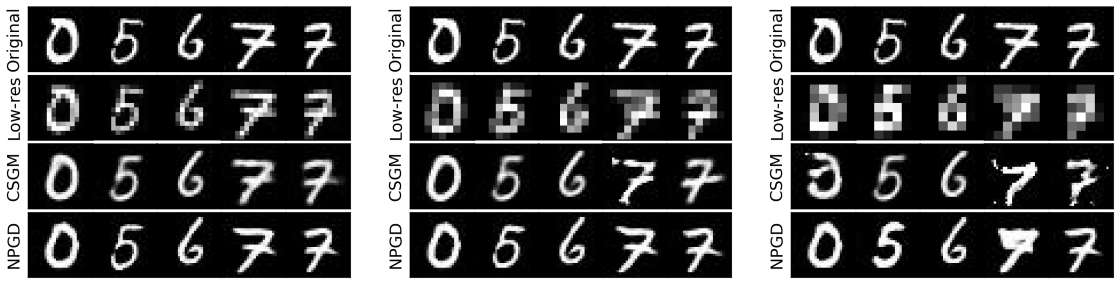
2.6.4 Comparison of Run-Time for Recovery

Table 2.1: Comparison of execution time ([sec.]) of recovery algorithms on the CelebA dataset. The relative speedup of our NPGD over the CSGM algorithm of Bora et al. is shown in parenthesis.

m	CSGM [1]	PGD-GAN	NPGD
200	5.8	66	0.09 (64x)
500	6.6	60	0.10 (66x)
1000	8.0	63	0.11 (72x)
2000	11.2	61	0.14 (80x)

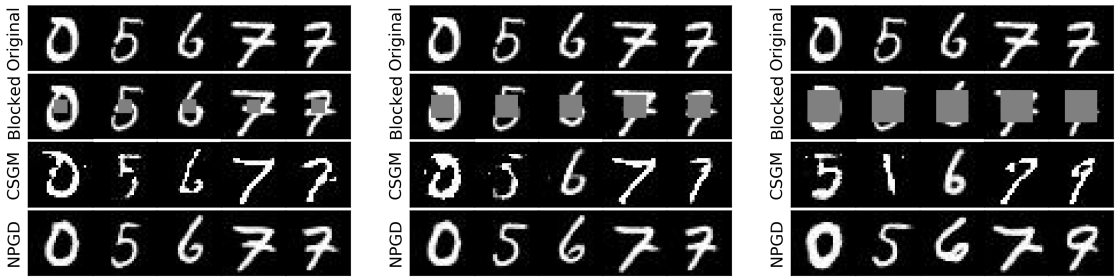
Table 2.1 compares the run times of our network-based algorithm NPGD and other recovery algorithms. We record the average run time to recover a single image from

[1]Run time includes 2 initializations, as implemented by the authors, for CelebA. The same number of initializations for CelebA (and 10 for MNIST) has been used to produce results in Fig. 2.6, 2.7, 2.8, and 2.10. Our NPGD algorithm uses only one, deterministic initialization, $x_0 = A^T y$.



(a) 4× low-res (b) 9× low-res (c) 16× low-res

Figure 2.10: Super-resolution on MNIST dataset. Row 1: original image x. Row 2: low-resolution images y, upsampled using constant padding, Row 3: high resolution image recovered by [28]. Row 4: high-resolution image recovered by our method.



(a) Mask size = 6 (b) Mask size = 10 (c) Mask size = 14

Figure 2.11: Inpainting in MNIST dataset. Row 1: original image x. Row 2: image y with center block missing. Row 3: image recovered by [28]. Row 4: image recovered by our method.

its compressed sensing measurements over 10 different images. All three algorithms were run on the same workstation with Intel i7-4770K CPU, 32GB RAM, and Nvidia GeForce Titan X GPU.

2.6.5 Analysis: Error in Projector

Figure 2.12 illustrates the idempotence error of the projector for different k. Three different categories of images are tested, namely, MNIST training samples, MNIST

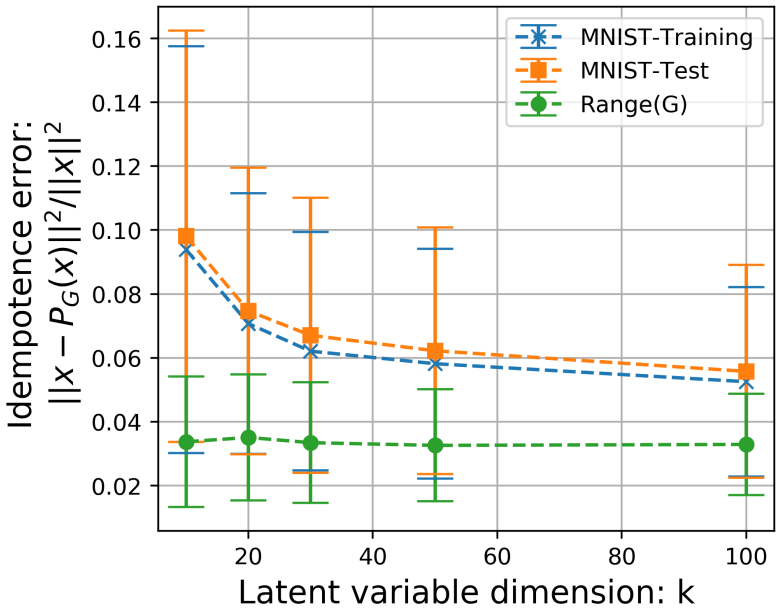


Figure 2.12: Idempotence error.

test samples, and samples $G(z)$ generated using the pre-trained G. We use clean images from the three sources and plot the relative idempotence error $\|x - P_G(x)\|^2/\|x\|^2$. The error decreases with increasing k and saturates around $k = 100$. The idemopotence errors for MNIST training and test samples are very close, indicating negligible generalization error. On the other hand, samples generated by $G(z)$ give much lower errors, which indicates representation error in the GAN. Thus we expect that a more flexible generator (deeper network) will lead to a better projector on the actual dataset and hence improve performance.

2.6.6 Comparison between G-Based and E2E Projector

We did preliminary experiments for compressive sensing on the end-to-end (E2E) projector trained using a fixed discriminator D on the MNIST dataset for exactly the same set-up as described in Section 2.6.1. The E2E projector P_G has the same architecture and structure as the G-based projector, i.e., $P_G = GG^\dagger$ for fair compar-

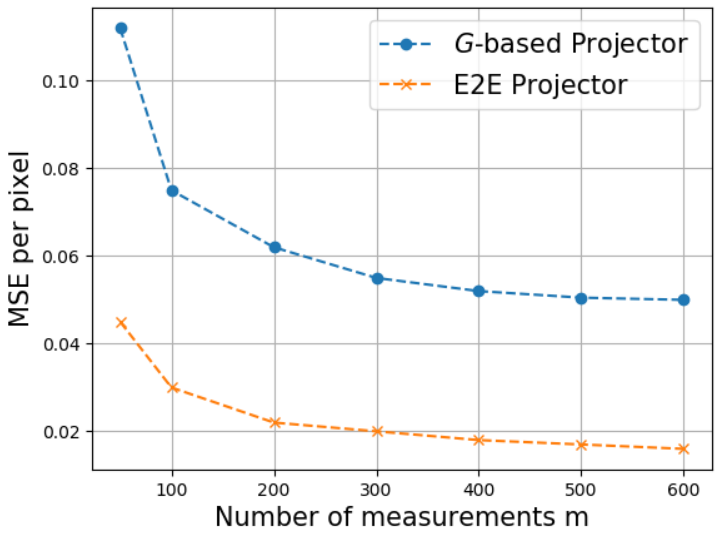


Figure 2.13: Comparison of reconstruction MSE loss per pixel for two projectors.

ison. Training of the network shown in the Figure 2.3 is done using the objective defined in (2.3), while keeping D fixed. We found that $\lambda = 1$ in (2.3), gave the best results.

Figure 2.13 shows the comparative results for reconstruction in compressive sensing with number of measurements $m = [50, 100, 200, 300, 400, 500, 600]$ on the x-axis. The y-axis shows the MSE per pixel i.e. $\|\hat{x} - x\|/n$ where x, $\hat{x}$ and n are clean image, reconstructed image and size of every image (28×28 for MNIST) respectively. As evident from the figure, the E2E projector outperforms the fixed G-based projector.

Chapter 3

ROBUSTNESS of INVERSE PROBLEMS

3.1 Motivation

Deep-learning-based methods for different applications have been shown vulnerable to adversarial examples. These examples make deployment of such models in safety-critical tasks questionable. The use of deep neural networks as inverse problem solvers has generated much excitement for medical imaging, e.g., magnetic resonance imaging (MRI), computed tomography (CT), etc., but recently a similar vulnerability has also been demonstrated for these tasks. Such applications demand the reconstruction to be stable and reliable. In this work, we propose methods for improving the robustness of deep-learning-based image reconstruction.

3.2 Problem Formulation

Antun et al. [43] identify instabilities of a deep-learning-based image reconstruction network by maximizing the following cost function:

$$Q_y(r) = \frac{1}{2}\|f(y + Ar) - x\|_2^2 - \frac{\lambda}{2}\|r\|^2 \tag{3.1}$$

As evident from this framework, the perturbation r is added in the x-space for each y, resulting in perturbation Ar in the y-space. We argue that this formulation can miss important aspects in image reconstruction, especially in ill-posed problems, for the following three main reasons:

1. It may not be able to model all possible perturbations to y. The perturbations $A\delta$ to y modeled in this formulation are all constrained to the range-space of

A. When A does not have full row rank, there exist perturbations to y that cannot be represented as $A\delta$.

2. It misses instabilities created by the ill-conditioning of the reconstruction problem. Consider a simple ill-conditioned reconstruction problem:

$$A = \begin{bmatrix} 1 & 0 \\ 0 & r \end{bmatrix} \text{ and } f = \begin{bmatrix} 1 & 0 \\ 0 & 1/r \end{bmatrix} \tag{3.2}$$

where A and f define the forward and reconstruction operator respectively, and $|r| \ll 1$. For $\delta = [0, \epsilon]^T$ perturbation in x, the reconstruction is $f(A(x+\delta)) = x + \delta$, and the reconstruction error is $\|f(A(x+\delta)) - x\|_2 = \epsilon$, that is, for small ϵ, the perturbation has negligible effect. In contrast, for the same perturbation δ in y, the reconstruction is $f(Ax+\delta) = x + [0, \epsilon/r]^T$, with reconstruction error $\|f(A(x+\delta)) - x\|_2 = \epsilon/r$, which can be arbitrarily large if $r \to 0$. This aspect is completely missed by the formulation based on (3.1).

3. For inverse problems, one also wants robustness to perturbations in the measurement matrix A. Suppose A used in training is slightly different from the actual $A' = A + \tilde{A}$ that generates the measurements. This results in perturbation $\tilde{A}x$ in y-space, which may be outside the range space of A, and therefore, as in 1 above, may not be possible to capture by the formulation based on (3.1).

The above points indicate that studying the problem of robustness to perturbations for image reconstruction problems in x-space misses possible perturbations in y-space that can have a huge adversarial effect on reconstruction. Since many of the image reconstruction problems are ill-posed or ill-conditioned, we formulate and study the issue of adversaries in the y-space, which is more generic and able to handle perturbations in the measurement operator A as well.

3.3 Image Reconstruction

Image reconstruction deals with recovering the clean image x from noisy and possibly incomplete measurements $y = Ax + v$. Recently, deep-learning-based approaches have outperformed the traditional techniques. Many deep learning architectures are inspired by iterative reconstruction schemes [1, 14, 28, 62]. Another popular way is to use an end-to-end deep network to solve the image reconstruction problem directly [45–49, 63, 64]. In this work, we propose modification in the training scheme for the end-to-end networks.

Consider the standard MSE loss in x-space with the popular ℓ_2-regularization on the weights (aka weight decay), which mitigates overfitting and helps in generalization [65]

$$\min_{\theta} \mathbb{E}_x \|f(Ax;\theta) - x\|^2 + \mu\|\theta\|^2 \tag{3.3}$$

In this paper, we experiment both with $\mu > 0$ (regularization present) and $\mu = 0$ (no regularization). No regularization is used in the sequel, unless stated otherwise.

3.4 Robustness Metric

We define a metric to compare the robustness of different networks. We measure the following quantity for network f:

$$\Delta_{\max}(x_0, \epsilon) = \max_{\|\delta\|_2 \leq \epsilon} \|f(Ax_0 + \delta) - x_0\|^2 \tag{3.4}$$

This determines the reconstruction error due to the worst-case additive perturbation over an ϵ-ball around the nominal measurement $y = Ax_0$ for each image x_0. The final robustness metric for f is $\rho(\epsilon) = \mathbb{E}_{x_0}[\Delta_{\max}(x_0, \epsilon)]$, which we estimate by the sample average of $\Delta_{\max}(x_0, \epsilon)$ over a test dataset,

$$\hat{\rho}(\epsilon) = \frac{1}{N} \sum_{i=1}^{N} \Delta_{\max}(x_i, \epsilon) \tag{3.5}$$

The smaller $\hat{\rho}$, the more robust the network.

We solve the optimization problem in (3.4) using projected gradient ascent (PGA) with momentum (with parameters selected empirically). Importantly, unlike training, where computation of $\Delta_{\max}(x_0)$ is required at every epoch, we need to solve (3.4) only once for every sample x_i in the test set, making this computation feasible during testing. To evaluate the effectiveness of PGA to find the attack, we plotted the evolution of $\hat{\rho}(\epsilon)$ over several iterations in Fig. 3.1. It indicates that the value of $\hat{\rho}$ increases with increasing iteration number and converges at $\sim 10^{th}$ iteration.

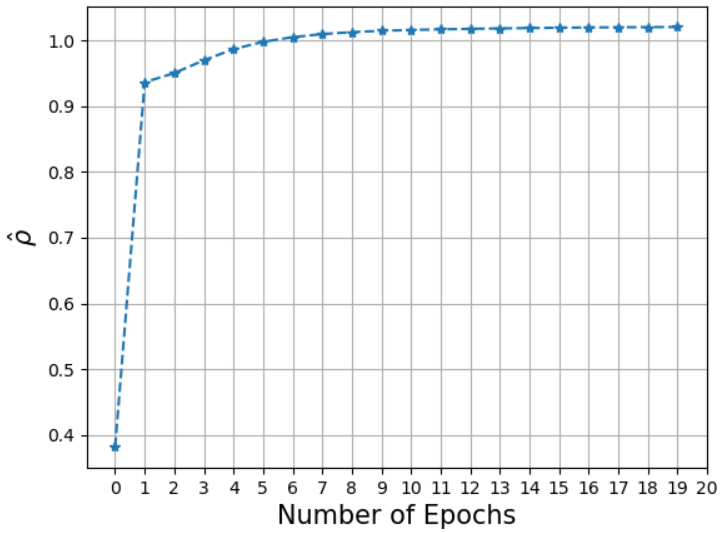


Figure 3.1: PGA evolution across iterations.

Chapter 4

ADVERSARIAL TRAINING

4.1 Background

One of the most powerful methods for training an adversarially robust network is adversarial training [35, 36, 66, 67]. It involves training the network using adversarial examples, enhancing the robustness of the network to attacks during inference. This strategy has been quite effective in classification settings, where the goal is to make the network output the correct label corresponding to the adversarial example.
Standard adversarial training involves solving the following min-max optimization problem:

$$\min_{\theta} \mathbb{E}_{(x,y)\in\mathbb{D}}[\max_{\delta:\|\delta\|_p\leq\epsilon} \mathcal{L}(f(x+\delta;\theta),y)] \tag{4.1}$$

where x is input sample to the network f parameterized by θ, y is the ground truth. $\mathcal{L}(\cdot)$ represents the applicable loss function, e.g., cross-entropy for classification, and δ is the perturbation added to each sample, within an ℓ_p-norm ball of radius ϵ.

This min-max formulation encompasses possible variants of adversarial training. It consists of solving two optimization problems: an inner maximization and an outer minimization problem. This corresponds to an adversarial game between the attacker and robust network f. The inner problem tries to find the optimal $\delta : \|\delta\|_p \leq \epsilon$ for a given data point (x, y) maximizing the loss, which essentially is the adversarial attack, whereas the outer problem aims to find a θ minimizing the same loss. For an optimal θ^* solving the Equation (4.1), then $f(;\theta^*)$ will be robust (in expected value) to all the x_{adv} lying in the ϵ-radius of ℓ_p-norm ball around the true x.

4.2 Adversarial Training for Image Reconstruction

Motivated by the adversarial training strategy (4.1), several frameworks have been proposed recently to make classification by deep networks more robust [31, 68, 69]. For image reconstruction, we propose to modify the training loss to the general form

$$\min_{\theta} \mathbb{E}_x \max_{\delta:\|\delta\|_p \leq \epsilon} \|f(Ax;\theta) - x\|^2 + \lambda \|f(Ax+\delta;\theta) - x\|^2 \tag{4.2}$$

The role of the first term is to ensure that the network f maps the non-adversarial measurement to the true x, while the role of the second term is to train f on worst-case adversarial examples within the ℓ_p-norm ball around the nominal measurement Ax. We want δ to be the worst-case perturbation for a given f. However, during the initial training epochs, f is mostly random (assuming random initialization of the weights) resulting in random perturbation, which makes f diverge. Hence we need only the first term during initial epochs to get a decent f that provides reasonable reconstruction. Then, reasonable perturbations are obtained by activating the second term, which results in robust f.

Now, solving the min-max problem above is intractable for a large dataset as it involves finding the adversarial example, which requires to solve the inner maximization for each $y = Ax$. This may be done using projected gradient descent (PGD), but is very costly. A possible sub-optimal approximation (with $p = 2$) for this formulation is:

$$\min_{\theta} \max_{\delta:\|\delta\|_2 \leq \epsilon} \mathbb{E}_x \|f(Ax;\theta) - x\|_2^2 + \lambda \|f(Ax+\delta;\theta) - x\|_2^2 \tag{4.3}$$

This formulation finds a common δ that is adversarial to each measurement y and tries to minimize the reconstruction loss for the adversarial examples together with that for clean examples. A similar approach by Madry et al. [35] is used to solve (4.1) in the classification setting, where a common δ is used for all the samples in the mini-batch and the classifier is trained to be robust to that δ. Now, for image reconstruction, clearly this is sub-optimal as using a perturbation δ common to all y's need not be the worst-case perturbation for any of the y's, and optimizing for the

common δ will not result in a highly robust network.

Ideally, we would want the best of both worlds: i.e., to generate δ for each y independently, together with tractable training. To this end, we propose to parameterize the worst-case perturbation $\delta = \arg\max_{\delta:\|\delta\|_2\leq\epsilon} \|f(y+\delta;\theta) - x\|_2^2$ by a deep neural network $G(y;\phi)$. This also eliminates the need of solving the inner-maximization to find δ using hand-designed methods. Since $G(\cdot)$ is parameterized by ϕ and takes y as input, a well-trained G will result in optimal perturbation for the given $y = Ax$. The modified loss function becomes:

$$\min_{\theta} \max_{\phi:\|G(\cdot,\phi)\|_2\leq\epsilon} \mathbb{E}_x \|f(Ax;\theta) - x\|^2 + \lambda \|f(Ax + G(Ax;\phi);\theta) - x\|^2 \tag{4.4}$$

This results in an adversarial game between the two networks: G and f, where G's goal is to generate strong adversarial examples that maximize the reconstruction loss for the given f, while f tries to make itself robust to the adversarial examples generated by the G. This framework is illustrated in the Fig. 4.1. This min-max setting is quite similar to the generative adversarial network (GAN), with the difference in the objective function. Also, here, the main goal is to build an adversarially robust f, which, to train, requires some empirical changes compared to the standard GANs (where the goal is to train a network to generate realistic samples from a distribution). This is explained in the next section. Another change is to reformulate the constraint $\|G(\cdot,\phi)\|_2 \leq \epsilon$ into a penalty form using the hinge loss, which makes the training more tractable:

$$\begin{aligned}\min_{\theta}\max_{\phi} \quad & \mathbb{E}_x \|f(Ax;\theta) - x\|^2 \\ & + \lambda_1 \|f(Ax + G(Ax;\phi);\theta) - x\|^2 \\ & + \lambda_2 \max\{0, \|G(Ax;\phi)\|_2^2 - \epsilon^2\}\end{aligned} \tag{4.5}$$

Note that λ_2 must be negative to satisfy the required constraint $\|G(\cdot,\phi)\|_2 \leq \epsilon$.

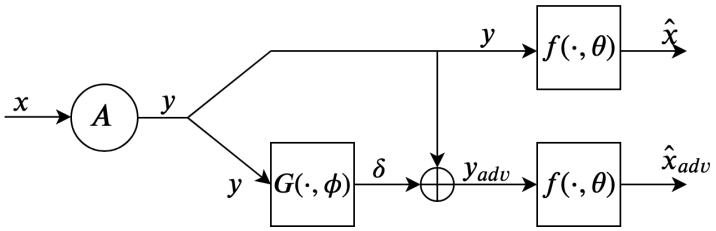


Figure 4.1: Adversarial training framework of image reconstruction network f, jointly with another network G, generating the additive perturbations.

4.2.1 Training Strategy

We apply some modifications and intuitive changes to train a robust f jointly with training G in a mini-batch set-up. At each iteration, we update G to generate adversarial examples and train f using those adversarial examples along with the non-adversarial or clean samples to make it robust. Along with the training of robust f, G is being trained to generate worst-case adversarial examples. To generate strong adversarial examples by G in the mini-batch update, we divide each mini-batch into K sets. Now, G is trained over each set independently and we use adversarial examples after the update of G for each set. This fine-tunes G for the small set to generate stronger perturbations for every image belonging to the set. Then, f is trained using the entire mini-batch at once but with the adversarial examples generated set-wise. G obtained after the update corresponding to the K^{th} set is passed for the next iteration or mini-batch update. This is described in Algorithm 2.

4.3 Theoretical Analysis

We theoretically obtained the optimal solution for the min-max formulation in (4.3) for a simple linear reconstruction. Although this analysis does not extend easily to the non-linear deep learning based reconstruction, it gives some insights for the behavior of the proposed formulation and how it depends on the conditioning of the measurement matrices.

Algorithm 2 Algorithm for training at iteration T

Input: Mini-batch samples (x_T, y_T), G_{T-1}, f_{T-1}
Output: G_T and f_T

$G_{T,0} = G_{T-1}$, $f = f_{T-1}$ Divide mini-batch into K parts.
while $k \leq K$ **do**
  $x = x_{T,k}, G = G_{T,k-1}$
  $G_{T,k} = \arg\max_G \lambda_1 \|f_{T-1}(Ax + G(Ax;\phi);\theta) - x\|^2 + \lambda_2 \max\{0, \|G(Ax;\phi)\|_2^2 - \epsilon^2\}$
  $\delta_{T,k} = G_{T,k}(x)$
end while
$\delta_T = [\delta_{T,1}, \delta_{T,2}, ..., \delta_{T,K}]$
$f_T = \arg\min_f \|f(Ax_T) - x_T\|^2 + \lambda_1 \|f(Ax_T + \delta_T) - x_T\|^2$
$G_T = G_{T,K}$
return G_T, f_T

Theorem 2 *Suppose that the reconstruction network f is a one-layer feed-forward network with no non-linearity i.e., $f = B$, where matrix B has SVD: $B = MQP^T$. Denote the SVD of the measurement matrix A by $A = USV^T$, where S is a diagonal matrix with singular values in permuted (increasing) order, and assume that the data is normalized, i.e., $E(x) = 0$ and $cov(x) = I$. Then the optimal $\hat{B}_0$ obtained by solving (4.3) is a modified pseudo-inverse of A, with $M = V$, $P = U$ and Q a filtered inverse of S, given by the diagonal matrix*

$$Q = \text{diag}\left(q_m, \ldots, q_m, 1/S_{m+1}, \ldots, 1/S_n\right),$$
$$q_m = \frac{\sum_{i=1}^m S_i}{\sum_{i=1}^m S_i^2 + \frac{\lambda}{1+\lambda}\epsilon^2} \tag{4.6}$$

with largest entry q_m of multiplicity m that depends on ϵ, λ and $\{S_i\}_{i=1}^n$.

Proof 2 *Please refer to the Appendix for the proof.*

The modified inverse B reduces the effect of ill-conditioning in A for adversarial cases in the reconstruction. This can be easily understood, using the simple example from the Equation (3.2). As explained previously, for the A in (3.2) with $|r| < 1$, an exact inverse, $f = \begin{bmatrix} 1 & 0 \\ 0 & \frac{1}{r} \end{bmatrix}$, amplifies the perturbation. Instead the min-max formulation

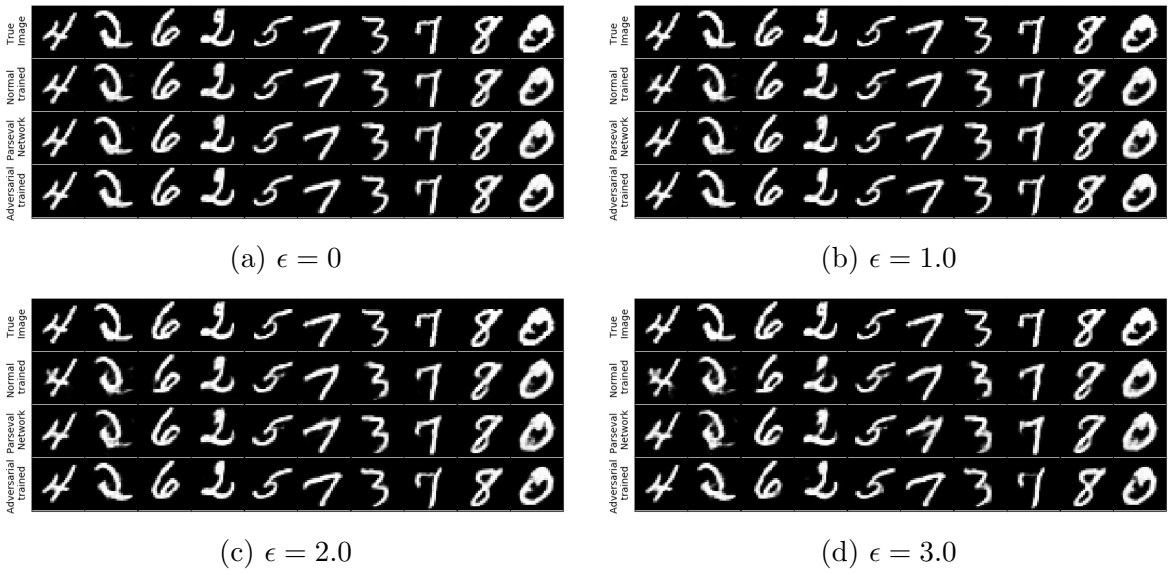
(4.3) (with $\lambda = 1$) results in a modified pseudo inverse $\hat{f} = \begin{bmatrix} 1 & 0 \\ 0 & \frac{r}{r^2+0.5\epsilon^2} \end{bmatrix}$, suppressing the effect of an adversarial perturbation $\delta = [0, \epsilon]^T$ in y as $\|f\delta\| \gg \|\hat{f}\delta\|$ for $r \to 0$ and $\epsilon \nrightarrow 0$. It can also be seen that $\hat{f}$ will not be optimal for the unperturbed y as it is not actual an inverse and the reconstruction loss using f for unperturbed case would be smaller than that for $\hat{f}$. However, for even very small adversaries, f would be much more sensitive than $\hat{f}$. It shows the trade-off between the perturbed and unperturbed case for the reconstruction in the case of ill-conditioned A.

This trade-off behavior will not manifest for a well-conditioned A, as an ideal linear inverse f for this case will not amplify the small perturbations and a reconstruction obtained using (4.3) with linear $\hat{f}$ will be very close to f (depending on ϵ): for well-conditioned A, $r \nrightarrow 0$. In that case $r^2 \gg 0.5\epsilon^2$, which reduces $\hat{f}$ to f.

Our experiments with deep-learning-based non-linear image reconstruction methods for CS using as sensing matrices random rows of a Gaussian matrix (well-conditioned) vs. random rows and columns of a DCT matrix (relatively ill-conditioned) indeed show the qualitatively different behavior with an increasing amount of perturbations.

4.4 Experiments and Results

Network Architecture: For the reconstruction network f, we follow the architecture of deep convolutional networks for image reconstruction. They use multiple convolution, deconvolution, and ReLU layers, and use batch normalization and dropout for better generalization. As a pre-processing step, which has been found to be effective for reconstruction, we apply the transpose (adjoint) of A to the measurement y, feeding $A^T y$ to the network. This transforms the measurement into the image-space, allowing the network to operate purely in image space. Since, f operates in the image domain, and f needs to learn the approximate inverse of $A^T A$, which often has shift-invariant structure, it is reasonable to use an architecture providing shift-invariance and capturing spatial correlation, motivating the choice of deep convolutional networks as architecture for f.



(a) $\epsilon = 0$ (b) $\epsilon = 1.0$

(c) $\epsilon = 2.0$ (d) $\epsilon = 3.0$

Figure 4.2: Qualitative Comparison for the MNIST dataset for different perturbations. *First row* of each sub-figure corresponds to the true image, *Second row* to the reconstruction using normally trained model, *Third row* to the reconstruction using Parseval network, *Fourth row* to the reconstruction using the adversarially trained model (***proposed scheme***).

Now, this spatial correlation and shift-invariance property are not true for the measurement y-space. Since, the adversarial perturbation generator G operates in the y-space, we use a standard feed-forward network for G with y as its input. The network consists of multiple fully-connected and ReLU layers. We trained the architecture shown in Fig. 4.1 using the objective defined in (4.5).

We designed networks of similar structure but a different number of layers for the two datasets, MNIST and CelebA used in the experiments.

We used the Adam Optimizer with $\beta_1 = 0.5$, $\beta_2 = 0.999$, a learning rate of 10^{-4} and a mini-batch size of 128, but divided into $K = 4$ parts during the update of G, described in the Algorithm 2. During training, the size ϵ of the perturbation has to be neither too big (affects performance on clean samples) nor too small (results in less robustness). We empirically picked $\epsilon = 2$ for MNIST and $\epsilon = 3$ for the CelebA

datasets. However, during testing, we evaluated $\hat{\rho}$, defined in (3.5) for different ϵ's (including those not used while training), to obtain a fair assessment of robustness. We compare the adversarially trained model using the min-max formulation defined in Objective (4.5), with three models trained using different training schemes:

1. Normally trained model with no regularization, i.e., $\mu = 0$ in (4.5).
2. ℓ_2-norm weight regularized model, using (3.3) with $\mu > 10^{-6}$ (aka weight decay), chosen empirically to avoid over-fitting and improve robustness and generalization of the network.
3. Lipschitz constant ($\mathcal{L}$)-constrained Parseval network [70]. The idea is to constrain the overall Lipschitz constant $\mathcal{L}$ of the network to be ≤ 1, by making $\mathcal{L}$ of every layer, ≤ 1. Motivated by the idea that regularizing the spectral norm of weight matrices could help in the context of robustness, this approach proposes to constrain the weight matrices to also be orthonormal, making them *Parseval tight frames*. Let S_{fc} and S_c define the set of indices for fully connected and convolutional layers respectively. The regularization term to penalize the deviation from the constraint is

$$\frac{\beta}{2}\Big(\sum_{i\in S_{fc}} \|W_i^T W_i - I_i\|_2^2 + \sum_{j\in S_c} \|\mathbf{W_j}^T\mathbf{W_j} - \frac{I_j}{k_j}\|_2^2\Big) \tag{4.7}$$

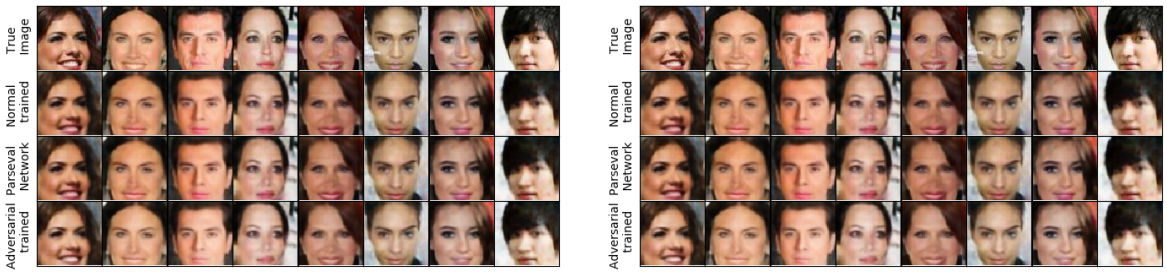
   where W_i is the weight matrix for ith fully connected layer and $\mathbf{W_j}$ is the transformed or unfolded weight matrix of jth convolution layer having kernel size k_j. This transformation requires input to the convolution to shift and repeat k_j^2 times. Hence, to maintain the *Parseval tight frames* constraint on the convolution operator, we need to make $\mathbf{W_j}^T\mathbf{W_j} \approx \frac{I_j}{k_j}$. I_i and I_j are identity matrices whose sizes depend on the size of W_i and $\mathbf{W_j}$ respectively. β controls the weight given to the regularization compared to the standard reconstruction loss. Higher β tends to deviate the network from predicting the true reconstruction while maintaining very close to Parseval tight frames and very low β does not provide any robustness and behaves like the network trained without any regularization. Empirically, we picked β to be 10^{-5}.

For a trained network, we checked the deviation of the learned weights of the network from the Parseval tight frames. To obtain the deviation, we computed the $D_j = \|\mathbf{W_j}^T\mathbf{W_j} - \frac{I_j}{k_j}\|_F$, where $\mathbf{W_j}$ is appropriately reshaped weight matrix, I_j is identity matrix and k_j is the kernel-size, D_j is the metric of deviation from the Parseval tight frame for the jth layer. For the Parseval network trained on MNIST dataset, we obtained $D_1 = 1.9e-4$, $D_2 = 1.015e-3$, $D_3 = 1.5e-3$ and $D_4 = 2.05e-3$. It shows that the training indeed results in the Parseval tight frames weights.

To compare different training schemes, we follow the same scheme (described below) for each dataset. Also, we extensively compare the performance for the two datasets for Compressive Sensing (CS) task using two matrices: one well-conditioned and another, relatively ill-conditioned. This comparison complements the theoretical analysis, discussed in the previous section.

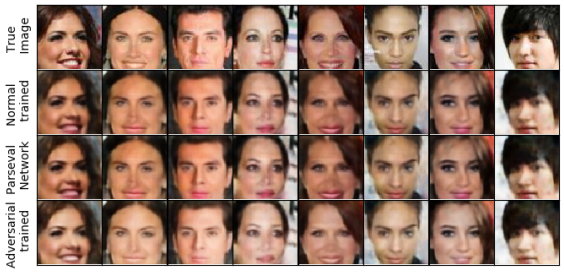
The **MNIST** dataset [58] consists of 28×28 gray-scale images of digits with $50,000$ training and $10,000$ test samples. The image reconstruction network consists of 4 convolution layers and 3 transposed convolution layers using re-scaled images between $[-1, 1]$. For the generator G, we used a network of 5 fully connected layers. Empirically, we found $\lambda_1 = 1$ and $\lambda_2 = -0.1$ in (4.5), gave the best performance in terms of robustness (lower $\hat{\rho}$) for different perturbations.

The **CelebA** dataset [59] consists of more than $200,000$ celebrity images. We use the aligned and cropped version, which pre-processes each image to a size of $64 \times 64 \times 3$ and scaled between $[-1, 1]$. We randomly pick $160,000$ images for the training. Images from the $40,000$ held-out set are used for evaluation. The image reconstruction network consists of 6 convolution layers and 4 transposed convolution layers. For the generator G, we used a network of 6 fully connected layers. We found $\lambda_1 = 3$ and $\lambda_2 = -1$ in (4.5) gave the best robustness performance (lower $\hat{\rho}$) for different perturbations.

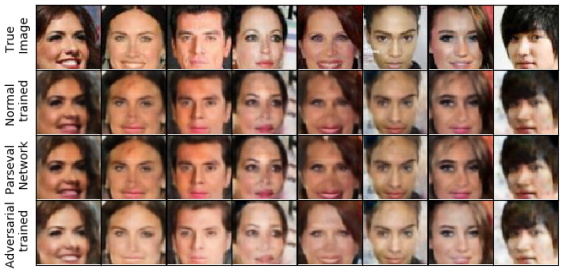


(a) $\epsilon = 0$

(b) $\epsilon = 2.0$



(c) $\epsilon = 5.0$



(d) $\epsilon = 10.0$

Figure 4.3: Qualitative Comparison for the CelebA dataset for different perturbations. *First row* of each sub-figure corresponds to the true image, *Second row* to the reconstruction using normally trained model, *Third row* to the reconstruction using Parseval network, *Fourth row* to the reconstruction using the adversarially trained model (***proposed scheme***).

4.4.1 Gaussian Measurement Matrix

In this set-up, we use the same measurement matrix A as recent works [1, 28], i.e. $A_{i,j} \sim N(0, 1/m)$ where m is the number of measurements. For MNIST, the measurement matrix $A \in R^{m \times 784}$, with $m = 100$, whereas for CelebA, $A \in R^{m \times 12288}$, with $m = 1000$. Figures 4.2 and 4.3 show the qualitative comparisons for the MNIST and CelebA reconstructions respectively, by solving the optimization described in Section 3.4. It can be seen clearly in both the cases that for different ϵ the adversarially trained models outperform the normally trained and Parseval networks. For higher ϵ's, the normally trained and Parseval models generate significant artifacts, which are much less for the adversarially trained models. Figures 4.4(a) and 4.4(b)

show this improvement in performance in terms of the quantitative metric $\hat{\rho}$, defined in (3.5) for the MNIST and CelebA datasets respectively. It can be seen that $\hat{\rho}$ is lower for the adversarially trained models compared to other training methods: no regularization, ℓ_2-norm regularization on weights, and Parseval networks (Lipschitz-constant-regularized) for different ϵ's, showing that adversarial training using the proposed min-max formulation indeed outperforms other approaches in terms of robustness. It is noteworthy that even for $\epsilon = 0$, adversarial training reduces the reconstruction loss, indicating that it acts as an excellent regularizer in general.

4.4.2 Discrete Cosine Transform (DCT) Matrix

To empirically study the effect of conditioning of the matrix, we did experiment by choosing A as random m rows and n columns of a $p \times p$ DCT matrix, where $p > n$. This makes A relatively more ill-conditioned than the random Gaussian A, i.e. the condition number for the random submatrix of the DCT matrix is higher than that of random Gaussian one. The number of measurements has been kept same as the previous case, i.e. $(m = 100,\ n = 784)$ for MNIST and $(m = 1000,\ n = 12288)$ for CelebA. We trained networks having the same configuration as the Gaussian ones. Figure 4.4 shows the comparison for the two measurement matrices. Based on the figure, we can see that $\hat{\rho}$ for the DCT, MNIST (Fig. 4.4(c)) and CelebA (Fig. 4.4(d)), are very close for models trained adversarially and using other schemes for the unperturbed case ($\epsilon = 0$), but the gap between them increases with increasing ϵ's, with adversarially trained models outperforming the other methods consistently. This behavior is qualitatively different from that for the Gaussian case (Fig. 4.4(a) and 4.4(b)), where the gap between adversarially trained networks and models trained using other (or no) regularizers is roughly constant for different ϵ.

4.4.3 Analysis with Respect to Conditioning

To check the conditioning, Fig. 4.5(a) shows the histogram for the singular values of the random Gaussian matrices. It can be seen that the condition number (ratio of

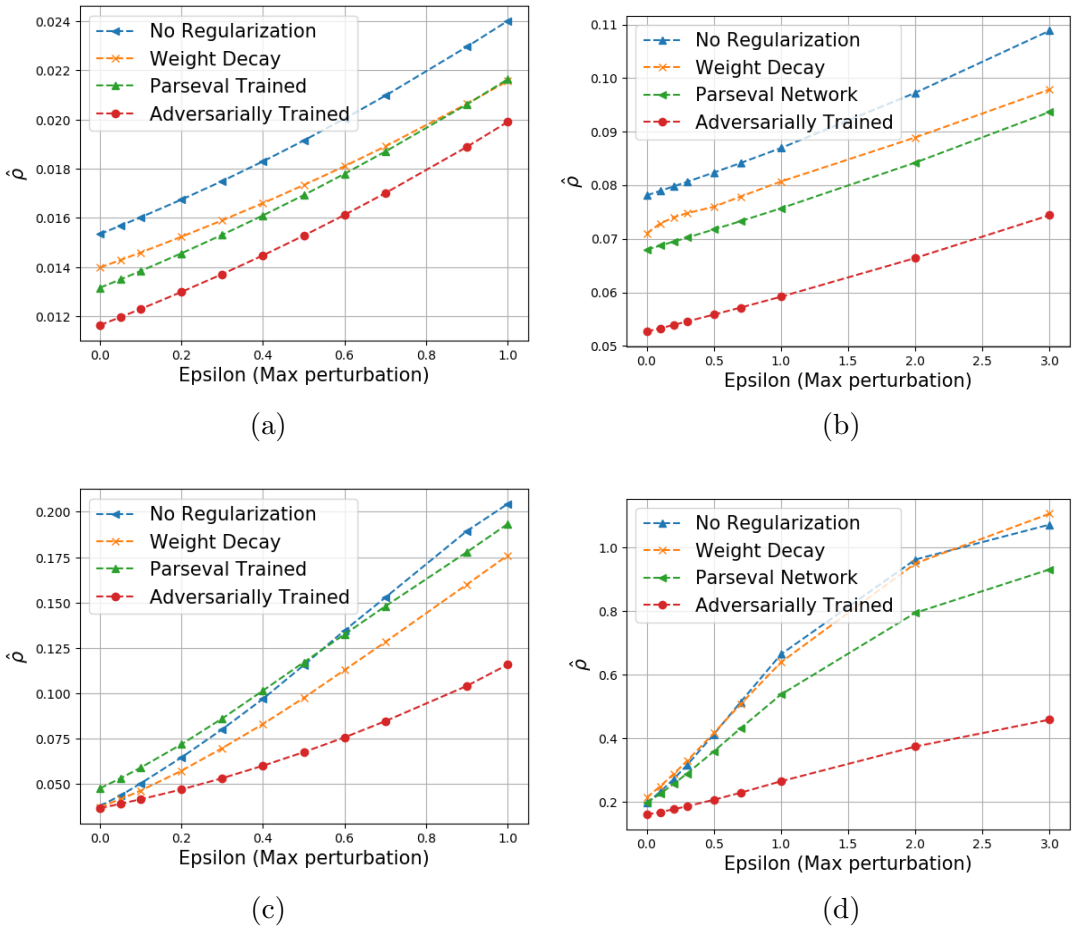
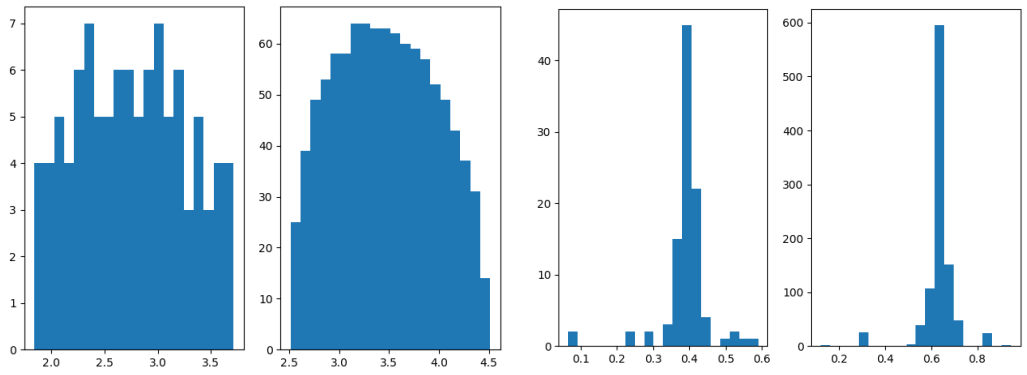


Figure 4.4: **Row 1** corresponds to the random rows of Gaussian measurement matrix: (a) MNIST, (b) CelebA. **Row 2** corresponds to random rows/columns of the DCT measurement matrix: (c) MNIST, (d) CelebA.



(a) (b)

Figure 4.5: (a) Distribution of the singular values for random rows of Gaussian measurement matrix: MNIST (left, $m = 100$) and CelebA (right, $m = 1000$) cases. (b) Distribution of the singular values for random rows/columns of DCT measurement matrix: MNIST (left, $m = 100$) and CelebA (right, $m = 1000$) cases.

maximum and minimum singular value) is close to 2 which is very well conditioned for both data sets. On the other hand, the histogram of the same for the random DCT submatrices (Fig. 4.5(b)) shows higher condition numbers – 8.9 for the 100×784 and 7.9 for the 1000×12288 dimension matrices, which is ill-conditioned relative to the Gaussian ones.

Referring to the above analysis of conditioning and plots of the robustness measure $\hat{\rho}$ for the two types of matrices: random Gaussian vs. random DCT indicate that the performance and behavior of the proposed min-max formulation depend on how well (or relatively ill)-conditioned the matrices are. This corroborates with the theoretical analysis for a simple reconstruction scheme (linear network) described in Sec. 4.3.

4.4.4 Linear Network for Reconstruction

We perform experiments using a linear reconstruction network in a simulated set-up to compare the theoretically obtained optimal robust reconstruction network with the one learned by our scheme. We take $50,000$ samples of a signal $x \in \mathbb{R}^{20}$ drawn

from $\mathcal{N}(0, I)$, hence, $\mathbb{E}(x) = 0$ and $cov(x) = I$. For the measurement matrix, we follow the same strategy as in Sec. 4.4.1, i.e. $\tilde{A}_{ij} \sim \mathcal{N}(0, 1/10)$. Since such matrices are well-conditioned, we replace 2 singular values of $\tilde{A}$ by small values (one being 10^{-3} and another, 10^{-4}) keeping other singular values and singular matrices fixed. This makes the modified matrix A ill-conditioned. We obtain the measurements $y = Ax \in \mathbb{R}^{10}$. For reconstruction, we build a linear network f having 1 fully connected layer with no non-linearity i.e. $f = B \in \mathbb{R}^{20\times 10}$.

Without Generator

In this case, we simulate exactly the same as theoretical set-up to obtain the reconstruction by optimizing the Objective (4.3) with linear $f = B$. The perturbation δ is directly obtained by optimizing the inner-max using the projected gradient descent (PGD), without using the attack generator. The reconstruction is given by $\hat{x} = \hat{B}_{PGD}y$, where $\hat{B}_{PGD}$ is obtained from:

$$\arg\min_{B} \max_{\delta:\|\delta\|_2\leq\epsilon} \mathbb{E}_x\|BAx - x\|^2 + \lambda\|B(Ax + \delta) - x\|^2 \tag{4.8}$$

We used $\lambda = 1$, $\epsilon = 0.1$, learning rate $= 0.001$ and momentum term as 0.9 in our experiments. We obtain the theoretically derived reconstruction $\hat{B}_0$ using the result given in (4.6) (from Theorem 2). To compare $\hat{B}_0$ and $\hat{B}_{PGD}$, we examined the following three metrics:

- $\|\hat{B}_{PGD} - \hat{B}_0\|_F/\|\hat{B}_0\|_F = 0.024, \|\hat{B}_{PGD} - \hat{B}_0\|_2/\|\hat{B}_0\|_2 = 0.034$
- $\|I - \hat{B}_0A\|_F/\|I - \hat{B}_{PGD}A\|_F = 0.99936$, where I is the identity matrix of size 20×20
- $\kappa(\hat{B}_0) = 19.231$, $\kappa(\hat{B}_{PGD}) = 19.311$, κ: condition number

The above three metrics indicate that when learned using Objective (4.8), $\hat{B}_{PGD}$ indeed converges to the solution $\hat{B}_0$ derived theoretically for the same objective.

With Generator

In this set-up, we use an attack generator G to obtain the perturbation δ using the objective defined in (4.5) with linear $f = B$. This leads to the two networks (G and $f = B$) pitting against each other, similar to experiments for deep networks. In this case, the perturbation is not necessarily same for all the measurement y, rather it depends on y and the parameters ϕ of G. The reconstruction is given by $\hat{x} = \hat{B}_{Gen}y$, where $\hat{B}_{Gen}$ is obtained from:

$$\begin{aligned}\hat{B}_{Gen} = \arg\min_{B}\max_{\phi} \quad & \mathbb{E}_x\|BAx - x\|^2 \\ & + \lambda_1\|B(Ax + G(Ax;\phi)) - x\|^2 \\ & + \lambda_2 \max\{0, \|G(Ax;\phi)\|_2^2 - \epsilon^2\} \end{aligned} \tag{4.9}$$

In our experiment, G is a feed forward network consisting of 4 fully connected layers with each layer followed by ReLU activation. Further, we used $\lambda_1 = 2$, $\lambda_2 = -0.05$, $\epsilon = 0.1$, learning rate $= 0.01$ as hyper-parameters to optimize (4.9) using the Adam optimizer. We obtain the theoretically derived reconstruction $\hat{B}_0$ using the result given in (4.6) (from Theorem 2). To compare $\hat{B}_0$ and $\hat{B}_{Gen}$, we examined the following three metrics:

- $\|\hat{B}_{Gen} - \hat{B}_0\|_F/\|\hat{B}_0\|_F = 0.0901$, $\|\hat{B}_{Gen} - \hat{B}_0\|_2/\|\hat{B}_0\|_2 = 0.102$
- $\|I - \hat{B}_0A\|_F/\|I - \hat{B}_{Gen}A\|_F = 0.997$, where I is the identity matrix of size 20×20
- $\kappa(\hat{B}_0) = 19.231$, $\kappa(\hat{B}_{Gen}) = 18.311$, κ: condition number

The results indicate that the min-max optimization scheme that we implemented with the nonlinear generator to solve Problem (4.9) is performing reasonably, producing a minimizer $\hat{B}_{Gen}$ close to the theoretical closed-form solution $\hat{B}_0$ produced as the solution to the related Problem (4.3). We also verified that the generator $G(y, \phi)$ with the learned ϕ in this case performs as expected, producing perturbations that depend on the specific measurement y. A somewhat unexpected empirical result is

that this did not lead to a significant change in the optimum robust linear reconstruction $\hat{B}_{Gen}$ from the theoretically derived solution to a somewhat different Problem (4.3), where the perturbation δ does not depend on the particular measurement y, but rather on the statistics of x. We conjecture that this is due to the solution $\hat{B}_0$ of Problem (4.3) coinciding with the solution $\hat{B}_1$ to Problem (4.2), in this very special case of linear reconstruction and $x \sim (0, \sigma^2 I)$. Now, because Problem (4.5) (taking the form (4.9) here) is set up to approximate Problem (4.2), we expect $\hat{B}_{Gen} \approx \hat{B}_1$. Hence by our conjecture that $\hat{B}_0 \approx \hat{B}_1$, we would expect $\hat{B}_{Gen} \approx \hat{B}_0$, which is what has been observed in this experiment. We leave the further study of these aspects to future work.

4.5 Modeling Perturbation Using G - Analysis

For a given reconstruction network f, the worst-case perturbation $\hat{\delta}$ for a sample input x, is given by:

$$\hat{\delta} = \underset{\delta:\|\delta\|_2 \leq \epsilon}{\arg\max} \|f(Ax + \delta) - x\|^2 \tag{4.10}$$

The above equation shows that $\hat{\delta}$ depends on Ax, x, hence we can write $\hat{\delta} = g(Ax, x)$ where $g(\cdot)$ is some function. It leads to three interesting observations related to the formulation of the attack generator G based on presumption that G has to learn to generate the worst-case perturbation for any given input x:

1. Assuming that G can indeed learn A implicitly (it will need to have a sufficiently expressive architecture), and Ax can then be computed from x, it is clear that one could generate $\hat{\delta} = g(Ax, x) = G(x, \phi)$.

2. If we formulate $\hat{\delta} = g(Ax, x) = G(Ax, x, \phi)$ i.e., we feed both the input x and the corresponding measurement y to the attack generator, $G(\cdot)$ needn't learn forward or inverse mapping implicitly, which are needed in the other two set-ups, and should result in stronger perturbation. This is an interesting future direction of how to design G which generates strong and diverse perturbations so that the reconstruction network f is robust to those perturbation.

3. Assuming that x can be recovered uniquely from $y = Ax$, say using the function f. Then it is also possible to generate $\hat{\delta} = g(Ax, x) = g(y, f(y)) = G(y, \phi)$. Now, the problem setup is predicated on the assumption that there exists θ such that $x \approx f(Ax, \theta)$. Hence, assuming that network G is expressive enough to generate internally the function f, then using $G(y, \phi)$ is justified.

We have used the formulation discussed in the 3^{rd} point above, but with modified goal and strategy. In Section 4.2.1, we discussed how at every iteration, we fine-tune the G for the small subset of measurements in the mini-batch to generate stronger perturbations for every image belonging to the set. This is to reduce the significance of expressiveness of the architecture of G. Because of this training strategy, we train G which generates a bad-case perturbation for every sample and because of fine-tuning every mini-batch, stronger perturbations are generated. The attack generator is only used during training, so as long as it can generate a good attack for each mini-batch, this is good enough, and we don't care that it does not generalize well, and hence we don't require quite an expressive G.

4.5.1 Evaluation of Trained G

To evaluate the performance of the trained attack generator G, we have compared the robustness metric $\hat{\rho}_{pgd}$ as defined in (3.5) obtained by solving (3.4) using PGD (which we consider the worst-case attack for given ϵ) with $\hat{\rho}_G$ obtained for the perturbation generated by G.

$$\hat{\rho}_G = \frac{1}{N} \sum_{i=1}^{N} \|f(Ax_i + G(Ax_i)) - x_i\|^2 \tag{4.11}$$

We defined another term to check if white Gaussian noise affects the network's robustness:

$$\hat{\rho}_v = \frac{1}{N} \sum_{i=1}^{N} \|f(Ax_i + v_i) - x_i\|^2 \text{ where } v_i \sim N(0, \epsilon^2 I) \tag{4.12}$$

Perturbations for all the three cases were scaled so that they have the same ℓ_2-norm, i.e. $\|G(Ax_i)\|_2 = \epsilon = 2.5$. Let $\hat{\rho}_0$ denotes the reconstruction loss for *no-perturbation*

case. The values obtained are:

$$\hat{\rho}_0 = \hat{\rho}_v = 0.38, \hat{\rho}_G = 0.8358, \hat{\rho}_{pgd} = 1.0205 \tag{4.13}$$

This indicates that random noise at the small level we consider here has no ill-effect on the reconstruction. Also, the trained attack generator is effective in generating bad-case perturbation but somewhat sub-optimal. As discussed earlier, slight sub-optimality is mitigated while training. Since training involves fine-tuning the attack generator every mini-batch, it becomes capable of generating a stronger perturbation specific to that mini-batch.

Chapter 5

INTERVAL-BOUND PROPAGATION

5.1 Motivation

Adversarial training requires solving two optimization problems simultaneously: inner maximization and outer minimization.

$$\min_{\theta} \mathbb{E}_x \max_{\delta:\|\delta\|_2\leq\epsilon} \|f(Ax;\theta) - x\|^2 + \lambda\|f(Ax+\delta;\theta) - x\|^2 \tag{5.1}$$

Min-max problems have saddle point(s) as optimal solution and are generally found to be difficult to solve. Furthermore, since gradient-based methods are popularly used to solve the inner-max i.e., $\max_{\delta:\|\delta\|_2\leq\epsilon} \|f(Ax+\delta;\theta) - x\|^2$, which is a non-convex problem, the result obtained is a lower bound on the max, rather than the true max. Minimization of a lower bound on the max, as in the adversarial training set-up, is sub-optimal because minimizing a lower bound of max does not guarantee mitigating effect of the worst-case perturbation. Here, we propose a method to obtain an upper bound on the inner max, whose minimization will account for handling the worst-case perturbation and can be therefore preferable to adversarial training. Even though the upper bound obtained is quite loose, it has significant computation advantage over adversarial training as it requires just two forward passes for the reconstruction network and does not require training of an additional generator network, as in the adversarial training set-up described in the previous chapter. The potential drawback is that the use of an upper bound may result in a conservative solution that would trade-off accuracy in the zero perturbation scenario for robustness.

5.2 Methodology

Since most deep networks can be written as composition of layers, we can obtain a bound on the output of the entire network by propagating the bound appropriately through individual layers. We describe how to obtain the bounds through each layer below.

5.2.1 Deep Networks

Deep neural networks (DNNs) can be considered as cascade of multiple functions which are referred as layers. Mathematically, it can be written as:

$$o_l = \phi(o_{l-1}), l = 1, ..., L \tag{5.2}$$

$$\phi(\cdot) = p \circ g \circ f \circ (\cdot) \tag{5.3}$$

where l indexes the layers of the network. The functions $p(\cdot), g(\cdot), f(\cdot)$ differ across the type of neural networks being used – each layer of a standard feed-forward network (FN) is a cascade of a fully connected affine transform, activation, and, optionally dropout and/or batch normalization, whereas, in convolutional neural networks (CNNs), each layer consist of convolution operation, activation, and, optionally dropout and/or batch normalization. Similarly, recurrent neural networks (RNNs) have recurrent connection, activation and optionally dropout in cascade for each layer. Propagation of a bound through a deep network can be obtained by propagating through a cascade of functions for each layer sequentially. We have performed our experiments on FNs and CNNs which required propagation of the bound through the corresponding layers, as described in the next section.

5.2.2 Bound Propagation

In this section, we will describe the bound propagation for the output of each function (popularly used in FNs and CNNs) [71]. Given the lower and upper bounds on

input, we need to calculate corresponding lower and upper bounds for the output. Subsequently, we use an underline ($\underline{\cdot}$) and overline ($\overline{\cdot}$) for the lower and upper bounds, respectively, for each term. For vectors x and y, we define inequality component-wise, i.e., $x \leq y \implies x_i \leq y_i \; \forall i$.

Fully Connected Affine Transform

An affine transform is given by:

$$z_k = W_k z_{k-1} + b_k \tag{5.4}$$

where z_k and z_{k-1} are the output and input for the k^{th} layer. W_k and b_k are the weight matrix and bias vector of appropriate dimension, the k^{th} layer parameters. We are given bounds on the input z_{k-1} i.e. $\underline{z}_{k-1} \leq z_{k-1} \leq \overline{z}_{k-1}$. To find a bound on the output z_k, given W_k, b_k and bounds on z_{k-1}, we define two auxiliary terms corresponding to output of $(k-1)^{th}$ layer (or input to k^{th} layer) as:

$$\mu_{k-1} \triangleq \frac{\overline{z}_{k-1} + \underline{z}_{k-1}}{2} \tag{5.5}$$

$$r_{k-1} \triangleq \frac{\overline{z}_{k-1} - \underline{z}_{k-1}}{2} \tag{5.6}$$

Using Equations (5.4), (5.5) and (5.6), the auxiliary terms for the k^{th} layer output are obtained as:

$$\mu_k = W_k \mu_{k-1} + b_k \tag{5.7}$$

$$r_k = |W_k| r_{k-1} \tag{5.8}$$

where $|W|$ is the matrix of absolute values of the entries of W, i.e., $|W|_{ij} = |W_{ij}|$. By using above two terms and the definition of the auxiliary terms, we obtain the

bounds on the output z_k:

$$\underline{z}_k = \mu_k - r_k \tag{5.9}$$

$$\overline{z}_k = \mu_k + r_k \tag{5.10}$$

The output z_k satisfies the inequality, $\underline{z}_k \leq z_k \leq \overline{z}_k$.

Convolution Operation

The operation in a convolution layer is given by:

$$z_k = w_k * z_{k-1} + b_k \tag{5.11}$$

where z_k and z_{k-1} are the output and input for the k^{th} layer. w_k and b_k are the convolution kernel and bias of appropriate dimension, the k^{th} layer parameters. $'*'$ represents standard multi-dimension convolution operation whose dimension depends on that of the input z_{k-1} and output z_k. The Equation (5.12) can be written in modified form (equivalent to the general affine transform form (5.4)) as:

$$z_k = \tilde{w}_k \tilde{z}_{k-1} + b_k \tag{5.12}$$

where $\tilde{w}_k$ is sparse matrix with appropriate repetition and unfolding of kernel w_k and $\tilde{z}_{k-1}$ is vectorized form of z_{k-1}. These standard transformations do not change the output z_k. This modified form enables the use of bound propagation similar to the affine transform using auxiliary terms μ_{k-1} and r_{k-1} using the bounds on the input z_{k-1} defined as in (5.5) and (5.6). Applying the inverse of the transformation required for converting convolution to standard affine-form, we can convert the modified affine form back to convolution to get the auxiliary terms for the k^{th} layer:

$$\mu_k = w_k * \mu_{k-1} + b_k \tag{5.13}$$

$$r_k = |w_k| * r_{k-1} \tag{5.14}$$

where $'*'$ represents the standard convolution operation. Using the above two equations, along with (5.9) and (5.10), we can obtain bounds on the output z_k.

Activation function

In a general set-up, the activation function can be written as:

$$h_k = \sigma(z_k) \tag{5.15}$$

where $\sigma(\cdot)$ is any activation function. Typically, in the deep learning community, $\tanh(\cdot)$, $\text{sigmoid}(\cdot)$, $\text{ReLU}(\cdot)$ or $\text{softmax}(\cdot)$ are popularly used as $\sigma(\cdot)$. Since, these activation functions are monotonic, propagating the bound is straight-forward given the bounds on the input, i.e. if $\underline{z}_k \leq z_k \leq \overline{z}_k$, the output satisfies the inequality:

$$\underline{h}_k = \sigma(\underline{z}_k) \leq h_k \leq \sigma(\overline{z}_k) = \overline{h}_k \tag{5.16}$$

Dropout

Dropout involves randomly dropping or activating a particular neuron using samples from Bernoulli distribution (with certain probability, also called dropout probability). In mathematical form, it is written like:

$$d_k = p_k \odot h_k \tag{5.17}$$

where p_k represents the element-wise dropout probability obtained by sampling from a Bernoulli distribution, which are either 0 or 1, determining which elements of h_k have to be dropped. Since, p_k is either 0 or 1, the bound on the d_k is given by:

$$\underline{d}_k = p_k \odot \underline{h}_k \leq d_k \leq p_k \odot \overline{h}_k = \overline{d}_k \tag{5.18}$$

Batch Normalization

Standard batch normalization (BN) for a particular mini-batch k is defined as:

$$\hat{x}_k = BN(x_k) = \gamma_k \odot \frac{x_k - \mu_k}{\sqrt{\sigma_k^2 + \epsilon}} + \beta_k \tag{5.19}$$

This is done element-wise (k indicates the index of the element) for vectors, with σ^2 and μ denoting the variance and mean of each element of x over the batch. Parameters γ_k and β_k are trainable and ϵ is a hyper-parameter. The parameters μ_k, σ_k, γ_k and β_k depend on the mini-batch only during training. During inference, population statistics of the entire training set are used for μ and σ, and γ and β are fixed. Each element of the vector x is applied through the normalization using the statistics.
BN for the lower and upper bound of the input x_k (assuming $\underline{x}_k \leq x_k \leq \overline{x}_k$)is given by:

$$BN(\underline{x}_k) = \gamma_k \odot \frac{\underline{x}_k - \mu_k}{\sqrt{\sigma_k^2 + \epsilon}} + \beta_k \tag{5.20}$$

$$BN(\overline{x}_k) = \gamma_k \odot \frac{\overline{x}_k - \mu_k}{\sqrt{\sigma_k^2 + \epsilon}} + \beta_k \tag{5.21}$$

As the bounds depend on the trainable parameter γ_k, BN may not be increasing, resulting in two cases:

1. If $\gamma_k \geq 0$, then $\underline{\hat{x}_k} = BN(\underline{x}_k)$ and $\overline{\hat{x}_k} = BN(\overline{x}_k)$.
2. If $\gamma_k < 0$, $\underline{\hat{x}_k} = BN(\overline{x}_k)$ and $\overline{\hat{x}_k} = BN(\underline{x}_k)$.

Although we only want to minimize the worst-case output perturbation during inference (when BN is simply a fixed affine transformation), we need to use the bounds during the training of the network, when the BN scale factor γ may change sign from one batch to another. Therefore, the bounds used in training the network must hold regardless of the sign of γ. Hence, we need to combine the two cases, which indeed

gives the true bound:

$$\underline{\hat{x}_k} = min(BN(\underline{x}_k), BN(\overline{x}_k))) \tag{5.22}$$

$$\overline{\hat{x}_k} = max(BN(\underline{x}_k), BN(\overline{x}_k))) \tag{5.23}$$

Since deep networks of interest (FNs or CNNs), referred as $\mathbf{f}(\cdot)$, use the above functions in a known order, we can easily propagate the bounds through these functions sequentially. Given the bound on the input y, we can obtain the bounds on $\mathbf{f}(y)$. To know the bounds on the input, we need to change the norm-space from ℓ_2 to ℓ_∞ for the constraints on δ in Equation (5.1).

5.2.3 Modified Loss Function

For any vector y of size m, we have the inequality:

$$\|y\|_2 \leq \sqrt{m}\|y\|_\infty \tag{5.24}$$

Using the above inequality, we modify (5.1) to use constraints in terms of ℓ_∞-norm $\|\delta\|_\infty \leq \epsilon'$ with $\epsilon' \triangleq \frac{\epsilon}{\sqrt{m}}$

$$\min_\theta \mathbb{E}_x \max_{\delta:\|\delta\|_\infty \leq \epsilon'} \|f(Ax;\theta) - x\|_2^2 + \lambda\|f(Ax+\delta;\theta) - x\|_2^2 \tag{5.25}$$

For $\|\delta\|_\infty \leq \epsilon'$, we know that: $Ax - \epsilon'\mathbf{1} \leq Ax + \delta \leq Ax + \epsilon'\mathbf{1}$, hence, we can rewrite (5.25) as:

$$\min_\theta \mathbb{E}_x \|f(Ax;\theta) - x\|_2^2 + \lambda\mathbb{E}_x \max_{Ax-\epsilon'\mathbf{1} \leq \alpha \leq Ax+\epsilon'\mathbf{1}} \|f(\alpha;\theta) - x\|_2^2 \tag{5.26}$$

where $\alpha = Ax + \delta$. Using the bound propagation discussed above, we can obtain a bound for the network f as:

$$\underline{f}(Ax, \epsilon', \theta) \leq f(Ax+\delta;\theta) \leq \overline{f}(Ax, \epsilon', \theta) \tag{5.27}$$

where $\underline{f}(Ax, \epsilon', \theta)$ and $\overline{f}(Ax, \epsilon', \theta)$ are the lower and upper bounds on $f(Ax + \delta; \theta)$ respectively, for $\|\delta\|_\infty \leq \epsilon'$. This bound on $f(\cdot)$ can be used to get a loose solution for the max-optimization in the (5.26) using the following inequality:

$$\|f(Ax + \delta; \theta) - x\|_2^2 \leq \sum_i max\left((\underline{f}(Ax, \epsilon', \theta)_i - x_i)^2, (\overline{f}(Ax, \epsilon', \theta)_i - x_i)^2\right) \quad (5.28)$$

The max above is element-wise maximum for each i, which is easy to obtain and does not require to solve any optimization problem. Combining the (5.26) and (5.28), we get the modified loss function as:

$$\min_\theta \mathbb{E}_x \|f(Ax; \theta) - x\|_2^2 + \lambda \mathbb{E}_x \sum_i max\left((\underline{f}(Ax, \epsilon', \theta)_i - x_i)^2, (\overline{f}(Ax, \epsilon', \theta)_i - x_i)^2\right) \quad (5.29)$$

This eliminates the need for an auxiliary network to solve the inner max in adversarial training, making this significantly faster. Also, since it computes an upper bound on the inner max compared to the lower bound (in our proposed adversarial training), it is potentially more effective in mitigating the worst-case perturbation effect. Minimizing an upper bound is not guaranteed to produce a better solution than minimizing a lower bound – depends on which one is tighter. But in this case, it is true that the minimized upper bound is a true upper bound on the output perturbation, so computing it can be used to provide a bound on sensitivity, at least for the training data. It will be interesting to do an extensive comparison and analysis of this technique with standard adversarial training discussed in chapter 4.

5.3 Experiments and Results

We performed preliminary experiments for compressive sensing on the MNIST dataset. The reconstruction network consists of 4 convolution and 3 transposed convolution layers along with batch normalization, dropout and ReLU. This architecture is exactly same as described in Section 4.4, used in adversarial training framework. The network is trained using the loss function defined in the Objective (5.29).

The bounds $\underline{f}$ and $\overline{f}$ on the output from the reconstruction network, are obtained by propagating the bound on the input through the network, as discussed in Section 5.2. The bound on the input, controls how much perturbation is allowed on the input, is defined by the hyper-parameter ϵ'. λ, the regularizer term, is another hyper-parameter which controls how much importance is given to the worst-case loss within the ϵ'-ball in ℓ_∞-norm around the input. Since, the bounds $\underline{f}$ and $\overline{f}$, are obtained for a fixed f, we need a *reasonably good* reconstruction f to begin with, for the IBP training; otherwise, the bounds will be very loose, making it difficult to train and converge.

In this set-up, we use the same measurement matrix A as in Section 4.4.1, i.e. $A_{i,j} \sim N(0, 1/m)$ where m is the number of measurements. For MNIST, the measurement matrix $A \in R^{m \times 784}$, with $m = 100$. For the IBP-based-training using Objective (5.29), the hyper-parameters λ and ϵ' were empirically picked as 1 and 0.2 respectively. During evaluation, we compute the robustness metric $\hat{\rho}$, defined in (3.5). The attacks for all the networks are obtained the same way i.e. solving (3.4) using the PGD algorithm. Figure 5.1 shows the comparative results of different algorithms used for training, discussed in detail in Section 4.4, with the IBP-based training. It can be seen from the figure that the IBP-based training outperforms the other techniques (results in the lowest reconstruction error) for different amounts of perturbation. Also, the IBP-based training provides a significant computational advantage. This training involves two forward passes through the network, whereas in adversarial training, we train another network in min-max formulation which involves back-propagation (which is expensive). In our experiments, we found that the IBP-based training is $\sim 3\times$ faster than our adversarial training method.

5.3.1 Discussion of the Zero-Perturbation Case

From Fig. 5.1, it is noteworthy that the network trained with *no regularization*, which explicitly minimizes the reconstruction error, does worse at zero-perturbation than other methods, which optimize a different loss function. It indicates that the regular-

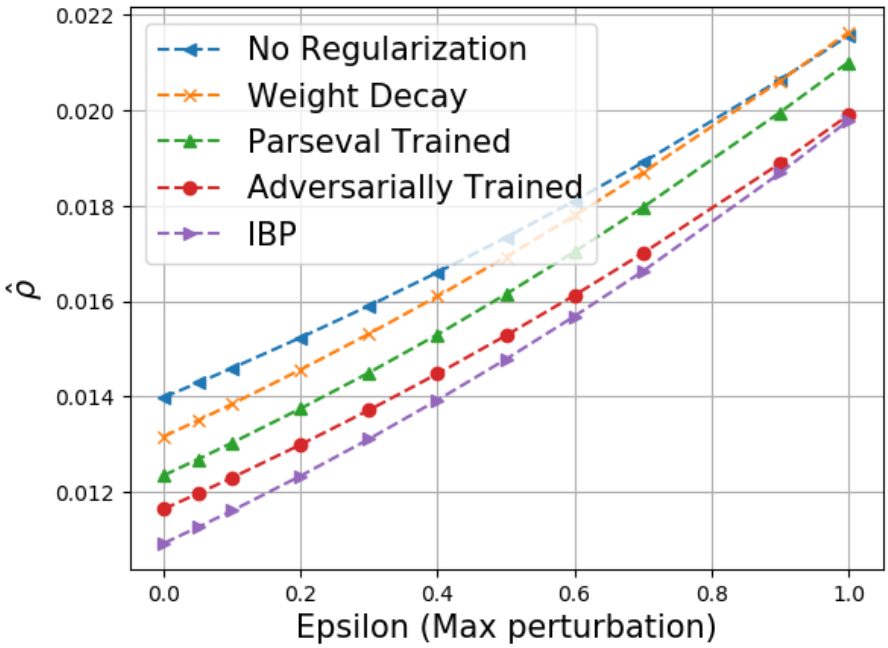


Figure 5.1: Comparison of different training algorithms in terms of robustness.

izers help in improving the generalization ability of the network along with improving the robustness. Since, evaluation is done on test dataset, the performance for zero-perturbation case, even for *no regularization* trained network, depends on how well the network generalizes, further validating the effectiveness of good regularizers in training deep networks.

Chapter 6

CONCLUSIONS

In this work, we addressed two important facets of deep-learning-based image reconstruction — *(i)* network design and guarantees, and *(ii)* robustness and stability of the system.

Firstly, we proposed a novel method of using GAN-based priors to solve ill-posed linear inverse problems using projected gradient descent (PGD). Empirically, our approach provides a speed-up of 60-80$\times$ over earlier GAN-based recovery methods with better accuracy. Our theoretical results show that for δ-approximate projector and moderately conditioned measurement matrix on the manifold range(G), the algorithm is guaranteed to reach $O(\delta)$ reconstruction error in $O(log(1/\delta))$ steps in low-noise regime.

Secondly, we argued that one should analyze and study the effect of adversaries and robustness in the measurement-space, instead of the signal-space, for inverse problems. We introduced an auxiliary network generating adversarial examples, used in min-max formulation to make image reconstruction networks robust. Theoretically, for a linear reconstruction scheme, we showed that the min-max formulation results in a singular-value(s) filter regularized solution, which suppresses the effect of adversarial examples occurring because of ill-conditioning in the measurement matrix.

Additionally, we proposed to use the idea of interval-bound propagation to minimize the upper bound on the reconstruction loss, given the perturbation. We showed that it is computationally more efficient and gives slightly better performance in terms of robustness than min-max formulation in our proposed adversarial training.

REFERENCES

[1] A. Raj, Y. Li, and Y. Bresler, "GAN-based projector for faster recovery with convergence guarantees in linear inverse problems," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 5602–5611.

[2] A. Raj, Y. Bresler, and B. Li, "Improving robustness of deep-learning-based image reconstruction," *arXiv preprint arXiv:2002.11821*, 2020.

[3] E. J. Candes, J. K. Romberg, and T. Tao, "Stable signal recovery from incomplete and inaccurate measurements," *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences*, vol. 59, no. 8, pp. 1207–1223, 2006.

[4] D. L. Donoho, "Compressed sensing," *IEEE Transactions on Information Theory*, vol. 52, no. 4, pp. 1289–1306, 2006.

[5] M. Elad and M. Aharon, "Image denoising via sparse and redundant representations over learned dictionaries," *IEEE Transactions on Image processing*, vol. 15, no. 12, pp. 3736–3745, 2006.

[6] W. Dong, L. Zhang, G. Shi, and X. Wu, "Image deblurring and super-resolution by adaptive sparse domain selection and adaptive regularization," *IEEE Transactions on Image Processing*, vol. 20, no. 7, pp. 1838–1857, 2011.

[7] B. Wen, S. Ravishankar, and Y. Bresler, "Structured overcomplete sparsifying transform learning with convergence guarantees and applications," *International Journal of Computer Vision*, vol. 114, no. 2-3, pp. 137–167, 2015.

[8] D. Liu, B. Wen, X. Liu, Z. Wang, and T. S. Huang, "When image denoising meets high-level vision tasks: A deep learning approach," *arXiv preprint arXiv:1706.04284*, 2017.

[9] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Bm3d image denoising with shape-adaptive principal component analysis," in *SPARS'09-Signal Processing with Adaptive Sparse Structured Representations*, 2009.

[10] J. Yang, J. Wright, T. S. Huang, and Y. Ma, "Image super-resolution via sparse representation," *IEEE Transactions on Image Processing*, vol. 19, no. 11, pp. 2861–2873, 2010.

[11] M. Elad, *Sparse and Redundant Representations: From Theory to Applications in Signal and Image Processing*. Springer Science & Business Media, 2010.

[12] C. Li, W. Yin, and Y. Zhang, "User's guide for TVAL3: TV minimization by augmented Lagrangian and alternating direction algorithms," *CAAM report*, vol. 20, no. 46-47, p. 4, 2009.

[13] S. V. Venkatakrishnan, C. A. Bouman, and B. Wohlberg, "Plug-and-play priors for model based reconstruction," in *Global Conference on Signal and Information Processing (GlobalSIP), 2013 IEEE*. IEEE, 2013, pp. 945–948.

[14] J. Rick Chang, C.-L. Li, B. Poczos, B. Vijaya Kumar, and A. C. Sankaranarayanan, "One network to solve them all–solving linear inverse problems using deep projection models," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5888–5897.

[15] J. Adler and O. Öktem, "Solving ill-posed inverse problems using iterative deep neural networks," *Inverse Problems*, vol. 33, no. 12, p. 124007, 2017.

[16] K. Fan, Q. Wei, L. Carin, and K. A. Heller, "An inner-loop free solution to inverse problems using deep neural networks," in *Advances in Neural Information Processing Systems*, 2017, pp. 2370–2380.

[17] H. Gupta, K. H. Jin, H. Q. Nguyen, M. T. McCann, and M. Unser, "CNN-based projected gradient descent for consistent CT image reconstruction," *IEEE Transactions on Medical Imaging*, vol. 37, no. 6, pp. 1440–1453, 2018.

[18] M. Mardani, Q. Sun, D. Donoho, V. Papyan, H. Monajemi, S. Vasanawala, and J. Pauly, "Neural proximal gradient descent for compressive imaging," in *Advances in Neural Information Processing Systems*, 2018, pp. 9596–9606.

[19] A. Mousavi, G. Dasarathy, and R. G. Baraniuk, "Deepcodec: Adaptive sensing and recovery via deep convolutional neural networks," *arXiv preprint arXiv:1707.03386*, 2017.

[20] A. Mousavi, G. Dasarathy, and R. G. Baraniuk, "A data-driven and distributed approach to sparse signal representation and recovery," in *7th International Conference on Learning Representations, ICLR 2019*, 2019.

[21] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems*, 2014, pp. 2672–2680.

[22] A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta, and A. A. Bharath, "Generative adversarial networks: An overview," *IEEE Signal Processing Magazine*, vol. 35, no. 1, pp. 53–65, 2018.

[23] J.-Y. Zhu, P. Krähenbühl, E. Shechtman, and A. A. Efros, "Generative visual manipulation on the natural image manifold," in *European Conference on Computer Vision*. Springer, 2016, pp. 597–613.

[24] R. A. Yeh, C. Chen, T. Yian Lim, A. G. Schwing, M. Hasegawa-Johnson, and M. N. Do, "Semantic image inpainting with deep generative models," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5485–5493.

[25] D. Berthelot, T. Schumm, and L. Metz, "Began: Boundary equilibrium generative adversarial networks," *arXiv preprint arXiv:1703.10717*, 2017.

[26] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang et al., "Photo-realistic single image super-resolution using a generative adversarial network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 4681–4690.

[27] M. Mardani, E. Gong, J. Y. Cheng, S. S. Vasanawala, G. Zaharchuk, L. Xing, and J. M. Pauly, "Deep generative adversarial neural networks for compressive sensing MRI," *IEEE Transactions on Medical Imaging*, vol. 38, no. 1, pp. 167–179, 2019.

[28] A. Bora, A. Jalal, E. Price, and A. G. Dimakis, "Compressed sensing using generative models," *arXiv preprint arXiv:1703.03208*, 2017.

[29] V. Shah and C. Hegde, "Solving linear inverse problems using GAN priors: An algorithm with provable guarantees," *arXiv preprint arXiv:1802.08406*, 2018.

[30] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013.

[31] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial machine learning at scale," *arXiv preprint arXiv:1611.01236*, 2016.

[32] M. Cisse, Y. Adi, N. Neverova, and J. Keshet, "Houdini: Fooling deep structured prediction models," *arXiv preprint arXiv:1707.05373*, 2017.

[33] K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, C. Xiao, A. Prakash, T. Kohno, and D. Song, "Robust physical-world attacks on deep learning models," *arXiv preprint arXiv:1707.08945*, 2017.

[34] C. Xiao, B. Li, J.-Y. Zhu, W. He, M. Liu, and D. Song, "Generating adversarial examples with adversarial networks," *arXiv preprint arXiv:1801.02610*, 2018.

[35] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," *arXiv preprint arXiv:1706.06083*, 2017.

[36] F. Tramèr, A. Kurakin, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel, "Ensemble adversarial training: Attacks and defenses," *arXiv preprint arXiv:1705.07204*, 2017.

[37] A. Athalye, N. Carlini, and D. Wagner, "Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples," *arXiv preprint arXiv:1802.00420*, 2018.

[38] E. Wong, F. Schmidt, J. H. Metzen, and J. Z. Kolter, "Scaling provable adversarial defenses," in *Advances in Neural Information Processing Systems*, 2018, pp. 8400–8409.

[39] Y. Jang, T. Zhao, S. Hong, and H. Lee, "Adversarial defense via learning to generate diverse attacks," in *The IEEE International Conference on Computer Vision (ICCV)*, October 2019.

[40] H. Jiang, Z. Chen, Y. Shi, B. Dai, and T. Zhao, "Learning to defense by learning to attack," *arXiv preprint arXiv:1811.01213*, 2018.

[41] W. Xu, D. Evans, and Y. Qi, "Feature squeezing: Detecting adversarial examples in deep neural networks," *arXiv preprint arXiv:1704.01155*, 2017.

[42] L. Schmidt, S. Santurkar, D. Tsipras, K. Talwar, and A. Madry, "Adversarially robust generalization requires more data," in *Advances in Neural Information Processing Systems*, 2018, pp. 5014–5026.

[43] V. Antun, F. Renna, C. Poon, B. Adcock, and A. C. Hansen, "On instabilities of deep learning in image reconstruction-does AI come at a cost?" *arXiv preprint arXiv:1902.05300*, 2019.

[44] J.-H. Choi, H. Zhang, J.-H. Kim, C.-J. Hsieh, and J.-S. Lee, "Evaluating robustness of deep image super-resolution against adversarial attacks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 303–311.

[45] B. Zhu, J. Z. Liu, S. F. Cauley, B. R. Rosen, and M. S. Rosen, "Image reconstruction by domain-transform manifold learning," *Nature*, vol. 555, no. 7697, p. 487, 2018.

[46] K. H. Jin, M. T. McCann, E. Froustey, and M. Unser, "Deep convolutional neural network for inverse problems in imaging," *IEEE Transactions on Image Processing*, vol. 26, no. 9, pp. 4509–4522, 2017.

[47] J. Schlemper, J. Caballero, J. V. Hajnal, A. Price, and D. Rueckert, "A deep cascade of convolutional neural networks for MR image reconstruction," in *International Conference on Information Processing in Medical Imaging*. Springer, 2017, pp. 647–658.

[48] G. Yang, S. Yu, H. Dong, G. Slabaugh, P. L. Dragotti, X. Ye, F. Liu, S. Arridge, J. Keegan, Y. Guo et al., "DAGAN: deep de-aliasing generative adversarial networks for fast compressed sensing MRI reconstruction," *IEEE Transactions on Medical Imaging*, vol. 37, no. 6, pp. 1310–1321, 2017.

[49] K. Hammernik, T. Klatzer, E. Kobler, M. P. Recht, D. K. Sodickson, T. Pock, and F. Knoll, "Learning a variational network for reconstruction of accelerated MRI data," *Magnetic Resonance in Medicine*, vol. 79, no. 6, pp. 3055–3071, 2018.

[50] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein GAN," *arXiv preprint arXiv:1701.07875*, 2017.

[51] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[52] R. G. Baraniuk and M. B. Wakin, "Random projections of smooth manifolds," *Foundations of Computational Mathematics*, vol. 9, no. 1, pp. 51–77, 2009.

[53] C. Hegde, A. C. Sankaranarayanan, W. Yin, and R. G. Baraniuk, "Numax: A convex approach for learning near-isometric linear embeddings," *IEEE Transactions on Signal Processing*, vol. 63, no. 22, pp. 6109–6121, Nov 2015.

[54] H. Kvinge, E. Farnell, M. Kirby, and C. Peterson, "A GPU-oriented algorithm design for secant-based dimensionality reduction," in *2018 17th International Symposium on Parallel and Distributed Computing (ISPDC)*. IEEE, 2018, pp. 69–76.

[55] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.

[56] H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena, "Self-attention generative adversarial networks," *arXiv preprint arXiv:1805.08318*, 2018.

[57] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, "Spectral normalization for generative adversarial networks," *arXiv preprint arXiv:1802.05957*, 2018.

[58] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[59] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild," in *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

[60] F. Yu, A. Seff, Y. Zhang, S. Song, T. Funkhouser, and J. Xiao, "LSUN: Construction of a large-scale image dataset using deep learning with humans in the loop," *arXiv preprint arXiv:1506.03365*, 2015.

[61] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 58, no. 1, pp. 267–288, 1996.

[62] B. Wen, S. Ravishankar, L. Pfister, and Y. Bresler, "Transform learning for magnetic resonance image reconstruction: From model-based learning to building neural networks," *arXiv preprint arXiv:1903.11431*, 2019.

[63] M. S. Sajjadi, B. Scholkopf, and M. Hirsch, "Enhancenet: Single image super-resolution through automated texture synthesis," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 4491–4500.

[64] H. Yao, F. Dai, S. Zhang, Y. Zhang, Q. Tian, and C. Xu, "Dr2-net: Deep residual reconstruction network for image compressive sensing," *Neurocomputing*, vol. 359, pp. 483–493, 2019.

[65] A. Krogh and J. A. Hertz, "A simple weight decay can improve generalization," in *Advances in Neural Information Processing Systems*, 1992, pp. 950–957.

[66] A. Sinha, H. Namkoong, and J. Duchi, "Certifying some distributional robustness with principled adversarial training," *arXiv preprint arXiv:1710.10571*, 2017.

[67] A. Arnab, O. Miksik, and P. H. Torr, "On the robustness of semantic segmentation models to adversarial attacks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 888–897.

[68] Y. Jang, T. Zhao, S. Hong, and H. Lee, "Adversarial defense via learning to generate diverse attacks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 2740–2749.

[69] H. Wang and C.-N. Yu, "A direct approach to robust deep learning using adversarial networks," *arXiv preprint arXiv:1905.09591*, 2019.

[70] M. Cisse, P. Bojanowski, E. Grave, Y. Dauphin, and N. Usunier, "Parseval networks: Improving robustness to adversarial examples," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 854–863.

[71] S. Gowal, K. Dvijotham, R. Stanforth, R. Bunel, C. Qin, J. Uesato, R. Arandjelovic, T. Mann, and P. Kohli, "On the effectiveness of interval bound propagation for training verifiably robust models," *arXiv preprint arXiv:1810.12715*, 2018.

Appendix A

PROOFS of THEOREMS 1 AND 2

A.1 Proof of Theorem 1

By the assumption of δ-approximate projection,

$$\|w_t - x_{t+1}\|^2 = \|w_t - G(G^\dagger(w_t))\|^2 \leq \|x^* - w_t\|^2 + \delta \tag{A.1}$$

where from the gradient update step, we have

$$w_t = x_t - \eta A^T(Ax_t - y) = x_t - \eta A^T A(x_t - x^*)$$

Substituting w_t into (A.1) yields

$$\begin{aligned}
&\|x_{t+1} - x_t\|^2 - 2\eta \left\langle x_{t+1} - x_t, A^T A\left(x^* - x_t\right)\right\rangle \\
&\leq \|x^* - x_t\|^2 - 2\eta\|A(x^* - x_t)\|^2 + \delta
\end{aligned}$$

Rearranging the terms we have

$$\begin{aligned}
&2\left\langle x_t - x_{t+1}, A^T A\left(x^* - x_t\right)\right\rangle \\
&\leq \frac{1}{\eta}\left\|x^* - x_t\right\|^2 - 2f\left(x_t\right) - \frac{1}{\eta}\left\|x_{t+1} - x_t\right\|^2 + \frac{\delta}{\eta} \\
&\leq \Big(\frac{1}{\eta\alpha} - 2\Big) f(x_t) - \frac{1}{\eta}\left\|x_{t+1} - x_t\right\|^2 + \frac{\delta}{\eta} \\
&\leq \Big(\frac{1}{\eta\alpha} - 2\Big) f\left(x_t\right) - \frac{1}{\eta\beta}\left\|Ax_{t+1} - Ax_t\right\|^2 + \frac{\delta}{\eta}
\end{aligned} \tag{A.2}$$

where the last two inequalities follow from $REC(S, \alpha, \beta)$. Now the LHS can be rewritten as:

$$\begin{aligned}
&2\left\langle x_t - x_{t+1}, A^T A\left(x^* - x_t\right)\right\rangle \\
&= \|Ax^* - Ax_{t+1}\|^2 - \|Ax^* - Ax_t\|^2 - \|Ax_{t+1} - Ax_t\|^2 \\
&= f(x_{t+1}) - f(x_t) - \|Ax_{t+1} - Ax_t\|^2 \qquad \text{(A.3)}
\end{aligned}$$

Combining (A.2) and (A.3), and rearranging the terms, we have:

$$f(x_{t+1}) \leq \Big(\frac{1}{\eta\alpha} - 1\Big) f(x_t) + \Big(1 - \frac{1}{\eta\beta}\Big) \|Ax_{t+1} - Ax_t\|_2^2 + \frac{\delta}{\eta}$$

and since $\eta = 1/\beta$,

$$f(x_{t+1}) \leq \Big(\frac{\beta}{\alpha} - 1\Big) f(x_t) + \beta\delta$$

For simplicity, we substitute $\kappa = \beta/\alpha$ in the following:

$$\begin{aligned}
f(x_n) &\leq (\kappa - 1)^n f\left(x_0\right) + \beta\delta \sum_{k=0}^{n-1} (\kappa - 1)^k \\
&= (\kappa - 1)^n f\left(x_0\right) + \frac{\beta\left(1 - (\kappa - 1)^n\right)}{2 - \kappa}\delta
\end{aligned}$$

For convergence, we require $1 \leq \kappa = \beta/\alpha < 2$. When n reaches $\frac{1}{2-\kappa} \log\left(\frac{f(x_0)}{C\alpha\delta}\right)$, we have

$$\begin{aligned}
&\|x_n - x^*\|^2 \leq \frac{\|Ax_n - Ax^*\|^2}{\alpha} = \frac{f(x_n)}{\alpha} \\
&\leq (\kappa - 1)^n \frac{f(x_0)}{\alpha} + \frac{\beta\left(1 - (\kappa - 1)^n\right)}{\alpha(2 - \kappa)}\delta \\
&\leq (\kappa - 1)^n \frac{f(x_0)}{\alpha} + \frac{\delta}{2/\kappa - 1} \leq \Big(C + \frac{1}{2/\kappa - 1}\Big)\delta
\end{aligned}$$

Finally, when $n \to \infty$, we have $(\kappa - 1)^n \frac{f(x_0)}{\alpha} \to 0$

$$\|x^* - x_\infty\|^2 \leq \frac{\delta}{2/\kappa - 1} = \frac{\delta}{2\alpha/\beta - 1}$$

A.2 Proof of Theorem 2

For the inverse problem of recovering the true x from the measurement $y = Ax$, the goal is to design a robust linear recovery model given by $\hat{x} = BAx$.

The min-max formulation to get robust model for a linear set-up:

$$\min_{B} \max_{\delta:\|\delta\|_2 \leq \epsilon} \mathbb{E}_{x \in D} \|BAx - x\|_2^2 + \lambda \|B(Ax + \delta) - x\|_2^2$$

$$\min_{B} \max_{\delta:\|\delta\|_2 \leq \epsilon} \mathbb{E}_{x \in D} (1+\lambda)\|BAx - x\|_2^2 + \lambda \|B\delta\|_2^2 + 2\lambda (B\delta)^T (BAx - x) \tag{A.4}$$

Since the data is normalized, we have $\mathbb{E}(x) = 0$ and $cov(x) = I$. This makes the above optimization problem as follows:

$$\min_{B} \max_{\delta:\|\delta\|_2 \leq \epsilon} \mathbb{E}_{x \in D} (1+\lambda)\|(BA - I)x\|_2^2 + \lambda \|B\delta\|_2^2$$

$$\min_{B} \max_{\delta:\|\delta\|_2 \leq \epsilon} \mathbb{E}_{x \in D} (1+\lambda) tr(BA - I)xx^T(BA - I)^T + \lambda \|B\delta\|_2^2 \tag{A.5}$$

Since $\mathbb{E}(tr(\cdot)) = tr(\mathbb{E}(\cdot))$, the above problem becomes:

$$\min_{B} \max_{\delta:\|\delta\|_2 \leq \epsilon} (1+\lambda) tr(BA - I)(BA - I)^T + \lambda \|B\delta\|_2^2$$

$$\min_{B} \max_{\delta:\|\delta\|_2 \leq \epsilon} (1+\lambda) \|BA - I\|_F^2 + \lambda \|B\delta\|_2^2 \tag{A.6}$$

Using SVD decomposition of $A = USV^T$ and $B = MQP^T$:

$$\min_{M,Q,P: M^TM = I, P^TP = I, Q \text{ is diag}} \max_{\delta:\|\delta\| \leq \epsilon} (1+\lambda) \|MQP^TUSV^T - I\|_F^2 + \lambda \|MQP^T\delta\|_2^2 \tag{A.7}$$

Since only the second term is dependent on δ, we can solve the inner maximum as follows:

We have $\|MQP^T\delta\|_2 = \|QP^T\delta\|_2$ since M is unitary. Given Q is diagonal, $\|QP^T\delta\|_2$ w.r.t. δ can be maximized by having $P^T\delta$ vector have all zeros except for the location corresponding to the $\max_i Q_i$. Since P is unitary, we have $\|P^T\delta\|_2 = \|\delta\|_2$. Hence, to maximize within the ϵ-ball, we will have $P^T\delta = \epsilon[0, .., 0, 1, 0, .., 0]$ where 1 is at the $\arg max_i Q_i$ position. This leads to the following result:

$$\max_{\delta:\|\delta\|_2\leq\epsilon} \|MQP^T\delta\|_2^2 = \epsilon^2(\max_i Q_i)^2$$

Substituting the above term in Equation (A.7):

$$\begin{aligned}
&\min_{M,Q,P:M^TM=I,P^TP=I,Q \text{ is diag}} (1+\lambda)\|MQP^TUSV^T - I\|_F^2 + \lambda\epsilon^2(\max_i Q_i)^2 \\
&\min_{M,Q,P:..} (1+\lambda)tr(MQP^TUSV^T - I)(MQP^TUSV^T - I)^T + \lambda\epsilon^2(\max_i Q_i)^2 \\
&\min_{M,Q,P:..} (1+\lambda)tr(MQP^TUS^2U^TPQM^T - 2MQP^TUSV^T + I) + \lambda\epsilon^2(\max_i Q_i)^2 \\
&\min_{M,Q,P:..} (1+\lambda)tr(P^TUS^2U^TPQ^2 - 2MQP^TSV^T + I) + \lambda\epsilon^2(\max_i Q_i)^2
\end{aligned} \tag{A.8}$$

For the above equation, only the second term depends on M, minimizing the second term w.r.t. M, keeping others fixed:

$$\min_{M:M^TM=I} tr(-2MQP^TUSV^T)$$

Since this is a linear program with the quadratic constraint, relaxing the constraint from $M^TM = I$ to $M^TM \leq I$, will not change the optimal point as the optimal point will always be at the boundary, i.e., $M^TM = I$. Relaxation of the constraint leads to the following program:

$$\min_{M:M^TM\leq I} tr(-2MQP^TUSV^T)$$

This relaxed form is a convex program. Introducing the Lagrange multiplier matrix K for the constraint:

$$\mathcal{L}(M, K) = tr(-2MQP^TUSV^T + K(M^TM - I)) \tag{A.9}$$

Substituting $G = QP^TUSV^T$ and using the stationarity of Lagrangian, we obtain:

$$\Delta L_M = M(K + K^T) - G^T = 0 \implies ML = G^T \text{ where } L = K + K^T$$

From primal feasibility, we have $M^TM \leq I$ and since optimal point is at the boundary, M will satisfy $M^TM = I$

Because the problem is convex, the local minimum is the global minimum which satisfies the two conditions: stationarity of Lagrangian ($ML = G^T$) and primal feasibility ($M^TM = I$). By the choice of $M = V$ and $L = SU^TPQ$, both these conditions are satisfied implying $M = V$ is the optimal point.

Substituting $M = V$ in Equation (A.8), we get:

$$\begin{aligned}
&\min_{Q,P:..}(1+\lambda)tr(P^TUS^2U^TPQ^2 - 2VQP^TUSV^T + I) + \lambda\epsilon^2(\max_i Q_i)^2 \\
&\min_{Q,P:..}(1+\lambda)tr(P^TUS^2U^TPQ^2 - 2QP^TUS + I) + \lambda\epsilon^2(\max_i Q_i)^2 \\
&\min_{Q,P:..}(1+\lambda)\|QP^TUS - I\|_F^2 + \lambda\epsilon^2(\max_i Q_i)^2
\end{aligned} \tag{A.10}$$

Denote the ith column of $C = U^TP$ by c_i and the entries in Q are in decreasing order (as entries in S are in increasing order) and the largest entry q_m in Q, has multiplicity m, the Equation (A.10) becomes:

$$\min_{C,Q}(1+\lambda)\sum_{i=1}^{m}\|q_mSc_i - e_i\|_2^2 + \lambda\epsilon^2q_m^2 + (1+\lambda)\sum_{i=m+1}^{n}\|q_iSc_i - e_i\|_2^2 \tag{A.11}$$

If we consider the last term i.e. $i > m$, it can be minimized by setting $c_i = e_i$ which is equivalent to choosing $P_i = U_i$ and $q_i = 1/s_i$. This makes the last term ($= 0$), using $h = \lambda\epsilon^2/(1+\lambda)$, making the Equation (A.11) to be:

$$\min_{C,Q} \sum_{i=1}^{m} (c_i^T S q_m^2 S c_i - 2 e_i^T q_m S c_i + e_i^T e_i) + h q_m^2$$

$$\min_{C,Q} q_m^2 (\sum_{i=1}^{m} c_i^T S^2 c_i + h) - 2 q_m \sum_{i=1}^{m} S_i C_{ii} + \sum_{i=1}^{m} e_i^T e_i$$

The above term is upward quadratic in q_m, hence minima w.r.t. q_m will occur at $q_m^* = \frac{\sum_{i=1}^{m} S_i C_{ii}}{(\sum_{i=1}^{m} c_i^T S^2 c_i + h)}$. This will make the quadratic term as $\sum_{i=1}^{m} e_i^T e_i - \frac{(\sum_{i=1}^{m} S_i C_{ii})^2}{(\sum_{i=1}^{m} c_i^T S^2 c_i + h)}$, which has to be minimized w.r.t. C:

$$\min_{C} \sum_{i=1}^{m} e_i^T e_i - \frac{(\sum_{i=1}^{m} S_i C_{ii})^2}{(\sum_{i=1}^{m} c_i^T S^2 c_i + h)}$$

$$\max_{C} \frac{(\sum_{i=1}^{m} S_i C_{ii})^2}{(\sum_{i=1}^{m} c_i^T S^2 c_i + h)}$$

$$\max_{C} \frac{(\sum_{i=1}^{m} S_i C_{ii})^2}{\sum_{i=1}^{m} S_i^2 C_{ii}^2 + \sum_{j \neq i} S_j^2 C_{ij}^2 + h} \tag{A.12}$$

Also, we know that $C = U^T P$ implying $C_{ij} = u_i^T p_j$ which makes $\|C_{ij}\|_2 \leq 1$ as $\|u_i\|_2 = \|p_j\|_2 = 1$. To maximize the term given by the Equation (A.12), we can minimize the denominator by setting the term $C_{ij} = 0$, which makes the matrix C diagonal.

Divide the matrix U and P into two parts: one corresponding to $i \leq m$ and another $i > m$, where i represents the column-index of $C = U^T P$.
Let $U = [U_1|U_2]$ and $P = [P_1|P_2]$. From above, we have $P_2 = U_2$ for $i > m$, making $P = [P_1|U_2]$.

$$U^T = \left[\frac{U_1^T}{U_2^T}\right] \text{ and } P = [P_1|U_2]$$
$$U^T P = \begin{bmatrix} U_1^T P_1 & U_1^T U_2 \\ U_2^T P_1 & U_2^T U_2 \end{bmatrix} = \begin{bmatrix} U_1^T P_1 & \mathbf{0} \\ U_2^T P_1 & I \end{bmatrix}$$

Since, $U^T P$ is diagonal, we have $U_2^T P_1 = \mathbf{0}$, $U_1^T P_1 = \Gamma$ where Γ is diagonal. Also, we have $P_1^T P_1 = I$. The only way to satisfy this would be to make $P_1 = U_1$ which makes $P = U$ and $C = I$. It also results in

$$q_m^* = \frac{\sum_{i=1}^m S_i}{\sum_{i=1}^m S_i^2 + h}, \text{ where, } h = \epsilon^2 \frac{\lambda}{1+\lambda} \tag{A.13}$$

Hence, the resulting B would be of the form MQP^T where:

$$M = V, P = U \text{ and }, \tag{A.14}$$
$$Q = \begin{bmatrix} q_m^* & 0 & ... & 0 \\ 0 & q_m^* & .. & 0 \\ : & : & : & : \\ : & : & : & : \\ 0 & ... & 1/s_{m+1} & .. \\ : & : & : & : \\ : & : & : & : \\ 0 & ... & 0 & 1/s_n \end{bmatrix} \tag{A.15}$$